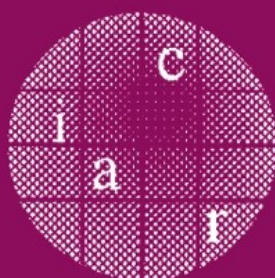


Chi Sung Laih (Ed.)

LNCS 2894

Advances in Cryptology – ASIACRYPT 2003

9th International Conference on the Theory
and Application of Cryptology and Information Security
Taipei, Taiwan, November/December 2003, Proceedings



Springer

Springer

Berlin

Heidelberg

New York

Hong Kong

London

Milan

Paris

Tokyo

Chi Sung Laih (Ed.)

Advances in Cryptology - ASIACRYPT 2003

9th International Conference on the Theory
and Application of Cryptology and Information Security
Taipei, Taiwan, November 30 – December 4, 2003
Proceedings



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editor

Chi Sung Laih
National Cheng Kung University
Department of Electrical Engineering
1 University Road, Tainan, Taiwan, R.O.C.
E-mail: laihs@eembox.ncku.edu.tw

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie;
detailed bibliographic data is available in the Internet at <<http://dnb.ddb.de>>.

CR Subject Classification (1998): E.3, D.4.6, K.6.5, F.2.1-2, C.2, J.1, G.2.2

ISSN 0302-9743

ISBN 3-540-20592-6 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag is a part of Springer Science+Business Media

springeronline.com

© International Association for Cryptologic Research 2003
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 10973370 06/3142 5 4 3 2 1 0

Preface

ASIACRYPT 2003 was held in Taipei, Taiwan, from Nov. 30 to Dec. 4, 2003. The 9th Annual ASIACRYPT conference was sponsored by the International Association for Cryptologic Research (IACR), this year in cooperation with the Chinese Cryptology and Information Security Association (CCISA) and National Cheng Kung University (NCKU) in Taiwan.

One hundred and eighty-eight papers from 26 countries were submitted to ASIACRYPT 2003 and 33 (of which one paper was withdrawn by the authors after notification) of these were selected for presentation. These proceedings contain revised versions of the accepted papers. We had an IACR 2003 Distinguished Lecture, by Dr. Don Coppersmith, entitled “Solving Low Degree Polynomials.” In addition, two invited talks were given at the conference. One was given by Dr. Adi Shamir. The other one was given by Dr. Hong-Sen Yan, entitled “The Secret and Beauty of Ancient Chinese Locks.” The conference program also included a rump session, chaired by Tzong Chen Wu, which featured short informal talks on recent results.

It was a pleasure for me to work with the program committee, which was composed of 27 members from 17 countries; I thank them for working very hard over several months. As a matter of fact, the review process was a challenging and time-consuming task, and it lasted about 8 weeks, followed by more than half a month for discussions among the program committee members. All submissions were anonymously reviewed by at least 3 members in the relevant areas of the program committee; in some cases, particularly for those papers submitted by a member of the program committee, they were reviewed by at least six members. We are grateful to all the program committee members who put in a lot of effort and precious time giving their expert analysis and comments on the submissions. In addition, we really appreciate the external referees who contributed with their expertise to the reviewing process; without their help, the selection process would not have gone so smoothly.

All paper submissions to ASIACRYPT 2003 were received electronically using the Web-based submission software, which was provided by Chanathip Namprempre. The review software was kindly provided by Bart Preneel, Wim Moreau, and Joris Claessens. I would like to thank Chien-Pang Kuo for his help with the installation and with solving problems we had with the software. I am also very grateful to Yi-Zhen Lin for her great help in handling ASIACRYPT 2003 affairs.

Special thanks to Yuliang Zheng, who acted as an advisory member of the committee and provided advice based on his previous experience. I would also like to thank the chair of IACR, Andy Clark, who gave me valuable advice on all kinds of problems.

For financial support of the conference, we are very grateful to this year’s sponsors, including the National Science Council, the Ministry of Education, the Directorate-General of Telecommunications, R.O.C., Chunghwa Telecom Co.,

Ltd., the Institute for Information Industry, Computer & Communications Research Labs, ITRI, etc.

Finally, we would like to thank all other people who provided any assistance, and all the authors who submitted their papers to ASIACRYPT 2003, as well as all the participants from all over the world.

September 2003

Chi Sung Laih

ASIACRYPT 2003

Nov. 30 – Dec. 4, 2003, Taipei, Taiwan

Sponsored by the
International Association for Cryptologic Research (IACR)

in cooperation with the
*Chinese Cryptography and Information Security Association,
National Cheng Kung University*

General Chair

Chin Chen Chang, National Chung Cheng University, No. 160, Sanshing Tsuen,
Minshiung Shiang, Chiai, Taiwan 621, Taiwan

Program Chair

Chi Sung Lai, Department of Electrical Engineering, National Cheng Kung
University, Tainan 701, Taiwan

Program Committee

Masayuki Abe	NTT Laboratories, Japan
Josh Benaloh	Microsoft Research, USA
Colin Boyd	QUT, Australia
Christian Cachin	IBM Zurich, Switzerland
Ivan Damgaard	University of Aarhus, Denmark
Robert H. Deng	Mui Keng Terrace, Singapore
Stefan Dziembowski	University of Warsaw, Poland
Matthias Fitzi	U.C. Davis, USA
Marc Joye	Gemplus, France
Kwangjo Kim	ICU, Korea
Pil Joong Lee	POSTECH, Korea
Chin Laung Lei	National Taiwan University, Taiwan
Arjen K. Lenstra	Citibank, USA
Tsutomu Matsumoto	Yokohama National University, Japan
Phong Q. Nguyen	ENS, France
Eiji Okamoto	University of Tsukuba, Japan
Carles Padró	Technical University of Catalonia, Spain
Si-han Qing	Chinese Academy of Sciences, China
Vincent Rijmen	KU Leuven, Belgium
Bimal Roy	Indian Statistical Institute, India
Reihaneh Safavi-Naini	University of Wollongong, Australia
Shiuh Pyng Shieh	National Chiao Tung University, Taiwan
Nigel P. Smart	University of Bristol, UK
Stefan Wolf	University of Montreal, Canada
Guozhen Xiao	Xidan University, China
Moti Yung	Columbia University, USA

Local Organizing Committee

Jinn-Ke Jan	Hsiang-Ling Chen
Wen-Guey Tzeng	Hui-Wen Du
Shiuh-Jeng Wang	Chien-Pang Kuo
Tzong-Chen Wu	Yi-Zhen Lin

Sponsors

National Science Council, Taiwan
Ministry of Education, Taiwan
Directorate General of Telecommunications,
Ministry of Transportation and Communications, Taiwan
Chunghwa Telecom Co., Ltd.
Institute for Information Industry
Computer & Communications Research Labs, ITRI

External Referees

Kazumaro Aoki	Min-Shiang Hwang	Louis Salvail
Feng Bao	Ren-Junn Hwang	Taiichi Saitoh
Steve Babbage	Shin-Jia Hwang	Palash Sarkar
Michael Backes	Yong Ho Hwang	Takakazu Satoh
Paulo Barreto	Albert Jeng	Berry Schoenmakers
Alexandre Benoit	Ji Hyun Jeong	Jong Hoon Shin
Eli Biham	Jorge Jim	Mahoro Shimura
Eric Brier	Qingguang Ji	Sang Gyoo Sim
Jan Camenisch	Jiménez Urroz Jorge	Leonie Simpson
Dario Catalano	Wen-Shenq Juang	Martijn Stam
Sanjit Chatterjee	Naoki Kanayama	Doug Stinson
Jiun-Ming Chen	Rajeeva L. Karandikar	Hung-Min Sun
Xiaofeng Chen	Chong Hee Kim	Koutarou Suzuki
Sandeepan Chowdhury	Ki Hyun Kim	Willy Susilo
Jean-Sébastien Coron	Tetsutaro Kobayashi	Alexei Tchoulkine
Coron Claude Crepeau	Hartono Kurnio	Dong To
Paolo D'Arco	Tanja Lange	Eran Tromer
Simon Pierre Desrosiers	John Malone-Lee	Wen-Guey Tzeng
Yvo Desmedt	C.H. Lin	Shigenori Uchiyama
Jean-François Dhem	Kai-Yung Lin	Frederik Vercauteren
Jeroen Doumen	Chi-Jen Lu	Jorge Luis Villar
Dang Nguyen Duc	E.H. Lu	Samuel S. Wagstaff
Orr Dunkelman	Anna Lysyanskaya	Shiuh-Jeng Wang
Ratna Dutta	Gwenaëlle Martinet	Benne de Weger
Chun-I Fan	Kazuto Matsuo	Christopher Wolf
Serge Fehr	Wenbo Mao	Hongjun Wu
Jacques J.A. Fournier	Joydip Mitra	Tzong-Chen Wu
Pierre-Alain Fouque	Sebastià Martín-Molleví	Wen-ling Wu
David Galindo	Yi Mu	Ching-Nung Yang
Steven Galbraith	Sourav Mukopadhyay	K. Yang
Sugata Gangopadhyay	Mridul Nandi	Yeon Hyeong Yang
Juan Gonzalez	Khanh Nguyen	Hsu-Chun Yen
Louis Granboulan	Miyako Ohkubo	Sung-Ming Yen
D.J. Guan	Daniel Page	Her-Tyan Yeh
Kishan Chand Gupta	Pascal Paillier	Yi-Shiung Yeh
Goichiro Hanaoka	Dong Jin Park	Sung Ho Yoo
Helena Handschuh	Jae Hwan Park	Young Tae Youn
Sang Yun Han	In Kook Park	Dae Hyun Yum
Keith Harrison	Joon Hah Park	Fangguo Zhang
Florian Hess	Jacques Patarin	Wentao Zhang
Javier Herranz	Duong Hieu Phan	Yuliang Zhen
Martin Hirt	Angela Piper	Huafei Zhu
Yvonne Hitchcock	Krzysztof Pietrzak	YongBin Zhou
Fumitaka Hoshino	Renato Renner	
Thomas Holenstein	Kouichi Sakurai	

Table of Contents

Public Key Cryptography I

Chosen-Ciphertext Security without Redundancy	1
<i>Duong Hieu Phan and David Pointcheval</i>	
Some RSA-Based Encryption Schemes with Tight Security Reduction	19
<i>Kaoru Kurosawa and Tsuyoshi Takagi</i>	
A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications	37
<i>Emmanuel Bresson, Dario Catalano, and David Pointcheval</i>	

Number Theory I

Factoring Estimates for a 1024-Bit RSA Modulus	55
<i>Arjen Lenstra, Eran Tromer, Adi Shamir, Wil Kortsmit, Bruce Dodson, James Hughes, and Paul Leyland</i>	
Index Calculus Attack for Hyperelliptic Curves of Small Genus	75
<i>Nicolas Thériault</i>	

Efficient Implementations

Parallelizing Explicit Formula for Arithmetic in the Jacobian of Hyperelliptic Curves	93
<i>Pradeep Kumar Mishra and Palash Sarkar</i>	
Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$	111
<i>Iwan Duursma and Hyang-Sook Lee</i>	
The AGM- $X_0(N)$ Heegner Point Lifting Algorithm and Elliptic Curve Point Counting	124
<i>David R. Kohel</i>	

Key Management and Protocols

Key Management Schemes for Stateless Receivers Based on Time Varying Heterogeneous Logical Key Hierarchy	137
<i>Miodrag J. Mihaljević</i>	
Leakage-Resilient Authenticated Key Establishment Protocols	155
<i>SeongHan Shin, Kazukuni Kobara, and Hideki Imai</i>	
Untraceable Fair Network Payment Protocols with Off-Line TTP	173
<i>Chih-Hung Wang</i>	

Hash Functions

Incremental Multiset Hash Functions and Their Application
to Memory Integrity Checking 188
*Dwaine Clarke, Srinivas Devadas, Marten van Dijk, Blaise Gassend,
and G. Edward Suh*

New Parallel Domain Extenders for UOWHF 208
*Wonil Lee, Donghoon Chang, Sangjin Lee, Soohak Sung,
and Mridul Nandi*

Cryptanalysis of 3-Pass HAVAL 228
Bart Van Rompay, Alex Biryukov, Bart Preneel, and Joos Vandewalle

Group Signatures

Efficient Group Signatures without Trapdoors 246
Giuseppe Ateniese and Breno de Medeiros

Accumulating Composites and Improved Group Signing 269
Gene Tsudik and Shouhuai Xu

Almost Uniform Density of Power Residues
and the Provable Security of ESIGN 287
Tatsuaki Okamoto and Jacques Stern

Number Theory II

Rotations and Translations of Number Field Sieve Polynomials 302
Jason E. Gower

On Class Group Computations Using the Number Field Sieve 311
Mark L. Bauer and Safuat Hamdy

Invited Talk

The Secret and Beauty of Ancient Chinese Padlocks 326
Hong-Sen Yan and Hsing-Hui Huang

Block Ciphers

A Traceable Block Cipher 331
Olivier Billet and Henri Gilbert

A New Attack against Khazad 347
Frédéric Muller

Broadcast and Multicast

An Efficient Public Key Trace and Revoke Scheme Secure
against Adaptive Chosen Ciphertext Attack 359
Chong Hee Kim, Yong Ho Hwang, and Pil Joong Lee

Sequential Key Derivation Patterns for Broadcast Encryption and Key Predistribution Schemes	374
<i>Nuttapong Attrapadung, Kazukuni Kobara, and Hideki Imai</i>	

Foundations and Complexity Theory

Boneh <i>et al.</i> 's k -Element Aggregate Extraction Assumption Is Equivalent to the Diffie-Hellman Assumption	392
<i>Jean-Sebastien Coron and David Naccache</i>	

On Diophantine Complexity and Statistical Zero-Knowledge Arguments ..	398
<i>Helger Lipmaa</i>	

Verifiable Homomorphic Oblivious Transfer and Private Equality Test	416
<i>Helger Lipmaa</i>	

Public Key Cryptography II

Generalized Powering Functions and Their Application to Digital Signatures	434
<i>Hisayoshi Sato, Tsuyoshi Takagi, Satoru Tezuka, and Kazuo Takaragi</i>	

Certificateless Public Key Cryptography	452
<i>Sattam S. Al-Riyami and Kenneth G. Paterson</i>	

A Complete and Explicit Security Reduction Algorithm for RSA-Based Cryptosystems	474
<i>Kaoru Kurosawa, Katja Schmidt-Samoa, and Tsuyoshi Takagi</i>	

The Insecurity of Esign in Practical Implementations	492
<i>Pierre-Alain Fouque, Nick Howgrave-Graham, Gwenaëlle Martinet, and Guillaume Poupard</i>	

Digital Signature

Efficient One-Time Proxy Signatures	507
<i>Huaxiong Wang and Josef Pieprzyk</i>	

Universal Designated-Verifier Signatures	523
<i>Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk</i>	

Author Index	543
--------------------	-----

Chosen-Ciphertext Security without Redundancy

Duong Hieu Phan and David Pointcheval

École normale supérieure – Dépt d’informatique
45 rue d’Ulm, 75230 Paris Cedex 05, France
{duong.hieu.phan,david.pointcheval}@ens.fr

Abstract. We propose asymmetric encryption schemes for which all ciphertexts are valid (which means here “reachable”: the encryption function is not only a probabilistic injection, but also a surjection). We thus introduce the Full-Domain Permutation encryption scheme which uses a random permutation. This is the first IND-CCA cryptosystem based on any trapdoor one-way permutation without redundancy, and more interestingly, the bandwidth is optimal: the ciphertext is over k more bits only than the plaintext, where 2^{-k} is the expected security level. Thereafter, we apply it into the random oracle model by instantiating the random permutation with a Feistel network construction, and thus using OAEP. Unfortunately, the usual 2-round OAEP does not seem to be provably secure, but a 3-round can be proved IND-CCA even without the usual redundancy $m||0^{k_1}$, under the partial-domain one-wayness of any trapdoor permutation. Although the bandwidth is not as good as in the random permutation model, absence of redundancy is quite new and interesting: many implementation risks are ruled out.

1 Introduction

By now, the widely admitted appropriate security level for asymmetric encryption is the so-called *chosen-ciphertext security* (IND-CCA): that is actually the semantic security [16] against adaptive chosen-ciphertext attacks [21]. For achieving semantic security, even in the basic chosen-plaintext scenario, the encryption algorithm must be probabilistic, which means that a given plaintext (with a fixed public key) should be possibly encrypted in many different ways (at least 2^k different ciphertexts if 2^{-k} is the expected security level). This naturally implies an expansion: the ciphertext is at least over k more bits than the plaintext. OAEP achieves the optimal bound if one considers IND-CPA only, but fails when considering IND-CCA [5, 15].

The general idea for designing cryptosystems which are secure in the sense of chosen-ciphertext security is indeed to make the decryption oracle useless by making the creation of new “valid” ciphertexts (which are not produced by actually encrypting some known plaintexts) impossible. The general approach is thus to add some redundancy either to the plaintext before encrypting [5] or in a tag appended to the ciphertext [4, 18]. The former method can be named

“encode-then-encrypt”, with a randomized bijective encoding (padding), and a trapdoor injective one-way function as encryption [5, 23, 8]. The latter is more like a key-encapsulation technique combined with a MAC of the plaintext, the ciphertext and/or the ephemeral key [10, 1, 18].

For symmetric encryption schemes, Desai [11] avoids the overhead due to the MAC or redundancy by using variable-length input PRF, variable-length output PRF (unbalanced Feistel paradigm) or variable-length input super-PRF (encode-then-encipher). The proposed schemes are chosen-ciphertext secure, without redundancy and the ciphertext expansion is smaller than for any other provably secure scheme.

In the present paper, inspired by this idea (encode-then-encipher), we consider the case of asymmetric encryption, by using a public random permutation which is clearly a bijective encoding, and this leads to the first IND-CCA scheme without any redundancy. More interestingly, the bandwidth of this scheme is optimal.

On the other hand, the security proof holds in the strong and ideal “random permutation model”. Such a scheme in a weaker model (the random oracle model or the standard model) would be better. The second part of this paper is devoted to this goal. We use the construction of OAEP but with 3 rounds, instead of 2, and we can prove that such a scheme is IND-CCA and all the ciphertexts are reachable by the encryption algorithm, and are thus valid (or almost all in the most general case).

The rest of the paper is organized as follows: We first briefly recall the security notions for asymmetric encryption; then we present the FDH encryption and we prove that it is IND-CCA secure with any trapdoor one-way permutation. Finally we consider the random oracle model, in which we propose a 3-round OAEP for which (almost) any ciphertext is valid (*i.e.*, reachable) and we show that it achieves IND-CCA under the partial-domain one-wayness of any trapdoor permutation [15].

2 Public Key Encryption

The aim of a public-key encryption scheme is to allow anybody who knows the public key of Alice to send her a message that she will be the only one able to recover, thanks to her private key.

2.1 Definitions

A public-key encryption scheme π is defined by the three following algorithms:

- The *key generation algorithm* \mathcal{G} . On input 1^k , where k is the security parameter, the algorithm \mathcal{G} produces a pair (pk, sk) of matching public and private keys.
- The *encryption algorithm* \mathcal{E} . Given a message m and a public key pk , $\mathcal{E}_{\text{pk}}(m)$ produces a ciphertext c of m . This algorithm may be probabilistic (involving random coins $r \in \mathcal{R}$, and then denoted $\mathcal{E}_{\text{pk}}(m; r)$.)
- The *decryption algorithm* \mathcal{D} . Given a ciphertext c and the secret key sk , $\mathcal{D}_{\text{sk}}(c)$ gives back the plaintext m .

2.2 Security Notions

The widely admitted security notion for encryption schemes is the so-called *semantic security* [16] (a.k.a. *polynomial security/indistinguishability of encryptions*): if the attacker has some *a priori* information about the plaintext, the view of the ciphertext should not increase this information. This security notion requires the computational impossibility to distinguish between two messages, chosen by the adversary itself, which one has been encrypted, with a probability significantly better than one half: its advantage $\text{Adv}_\pi^{\text{ind}}(\mathcal{A})$, as defined below where the adversary \mathcal{A} is seen as a 2-stage Turing machine (A_1, A_2) , should be negligible.

$$\text{Adv}_\pi^{\text{ind}}(\mathcal{A}) = 2 \times \Pr_{b,r} \left[(\text{pk}, \text{sk}) \leftarrow \mathcal{G}(1^k); (m_0, m_1, s) \leftarrow A_1(\text{pk}) \right. \\ \left. c = \mathcal{E}_{\text{pk}}(m_b; r) : A_2(m_0, m_1, s, c) = b \right] - 1.$$

Another notion has been thereafter defined, the so-called *non-malleability* [12], but this notion is equivalent to the above one in some specific scenarios [7]. Moreover, it is equivalent to the semantic security [3] in the most interesting scenarios, described below.

Indeed, an attacker can play many kinds of attacks: it may just have access to public data, and then encrypt any plaintext of its choice (*chosen-plaintext attacks*), or have access to extra information, modeled by various oracles. In this model, the strongest oracle is definitely the decryption algorithm, which can be queried on any ciphertext, except the challenge ciphertext (*adaptive/non-adaptive chosen-ciphertext attacks* [17, 21]).

A general study of these security notions and attacks has been driven in [3], we therefore refer the reader to this paper for more details. Actually, one conclusion is that the strongest security level is the so-called *chosen-ciphertext security*, which is the semantic security (IND) under adaptive chosen-ciphertext attacks (CCA), hence the notation IND-CCA, also known as IND-CCA2, to be compared to IND-CCA1, which captures lunchtime attacks [17] only.

2.3 Secure Designs

The expected security level is thus IND-CCA, which is now required to be provably achieved before any practical use. The last ten years have seen several practical proposals which provide this strong security level. The first, and most famous one, is definitely OAEP [5], a generic conversion proposed by Bellare and Rogaway, which applies to any trapdoor partial-domain one-way permutation, such as RSA, in the random oracle model [15]. Some variants have been recently proposed, which either apply to particular cases (SAEP, SAEP+ [8]) or more general ones (OAEP+ [23]). But they all add some redundancy in the plaintext before encrypting it: a ciphertext that is not properly generated, without knowing the plaintext, is valid with negligible probability only. The latter property had been formally defined by the *plaintext-awareness* notion [5, 3]. Granted it, a decryption oracle does not provide any information.

Some other paddings have also been proposed to apply to more general families of functions, which are not necessarily one-to-one: Fujisaki and Okamoto [13, 14], Pointcheval [20] and Okamoto and Pointcheval [18]. Once again, chosen-ciphertext security is achieved granted redundancy, but in the ciphertext: only properly generated ciphertexts (with some known plaintexts) have a chance to be valid: plaintext-awareness.

3 FDP: Full-Domain Permutation Encryption

In the same vein as the Full-Domain Hash signature [6, 9], we suggest the Full-Domain Permutation encryption, in which one applies a random permutation to the message (and the random coins) before encrypting it with the trapdoor one-way permutation. We therefore obtain the first cryptosystem which achieves chosen-ciphertext security, without redundancy: any ciphertext is valid, and the bandwidth is optimal.

3.1 Description

The FDP-encryption is quite simple, since it uses a random permutation \mathcal{P} (which is a bijective random oracle, or an ideal-cipher with a particular key, say 0. See also [22]). The key generation algorithm selects a trapdoor one-way permutation φ_{pk} (and its inverse ψ_{sk} , granted the trapdoor sk) over $\{0, 1\}^{k+\ell}$, and a random permutation \mathcal{P} over the same space — $\{0, 1\}^\ell \times \{0, 1\}^k$ is identified to $\{0, 1\}^{\ell+k}$. The public key pk thus defines the permutation φ_{pk} , while the private key sk defines the inverse ψ_{sk} of φ_{pk} . Then,

$$\mathcal{E}_{\text{pk}}(m; r) = \varphi_{\text{pk}}(\mathcal{P}(m, r)) \quad \mathcal{D}_{\text{sk}}(c) = m, \text{ where } (m, r) = \mathcal{P}^{-1}(\psi_{\text{sk}}(c)).$$

The space of the plaintexts is $\{0, 1\}^\ell$, while the space of the random coins r is $\{0, 1\}^k$. Note that both \mathcal{P} and \mathcal{P}^{-1} are public permutations.

Note that usual trapdoor one-way permutations are not on a binary set, as it will be discussed in a more extensive way in the following. Anyway, just doubling the computational cost, on average, one easily gets such a particular case from any permutation over an interval: [2] suggested an iterated version.

3.2 Security Result

As already said, the first advantage of this scheme is that any ciphertext is valid: any ciphertext can be decrypted into a plaintext, furthermore any ciphertext can also be reached by the encryption algorithm. The second important advantage comes from the security result given below: it provides chosen-ciphertext security under the intractability of inverting φ , with a security level in 2^k , with an overhead of k bits (the random coins). This means that the bandwidth is optimal: contrary to OAEP or OAEP+ which need an overhead of at least $2k$ bits (the random coins and the redundancy), for a similar security level. Of course, this remark only applies to the most general case where $\ell \geq k$ (e.g., $k = 80$ and $k + \ell = 1024$.)

Theorem 1. *Let \mathcal{A} be any chosen-ciphertext adversary against φ -FDP, within time τ . After q_p and q_d queries to the permutation oracles and the decryption oracle respectively,*

$$\text{Adv}_{\pi}^{\text{ind-cca}}(\mathcal{A}) \leq 2 \times \text{Succ}_{\varphi}^{\text{ow}}(\tau + 2q_p \times T_{\varphi}) + 2 \times \left(\frac{(q_p + q_d + 1)^2}{2^{k+\ell}} + \frac{q_p}{2^k} + \frac{(q_d + 1)^2}{2^{\ell}} \right)$$

where T_{φ} is the time complexity for evaluating φ .

Let us briefly recall that for any algorithm \mathcal{A} ,

$$\text{Succ}_{\varphi}^{\text{ow}}(\mathcal{A}) = \Pr_{\text{pk}} [\mathcal{A}(\varphi_{\text{pk}}(x)) = x], \text{ and } \text{Succ}_{\varphi}^{\text{ow}}(\tau) = \max_{|\mathcal{A}| \leq \tau} \{ \text{Succ}_{\varphi}^{\text{ow}}(\mathcal{A}) \}.$$

3.3 Sketch of the Proof

The goal of the proof is to simulate the oracles \mathcal{P} , \mathcal{P}^{-1} , and \mathcal{D}_{sk} in such a way that the adversary can not distinguish the simulations from the real oracles. In the simulation, the decryption answer for a ciphertext that has not been obtained before is a new random value (and independent with others). We then have to keep the simulation of the random permutation consistent. On the other hand, the challenge is made independent with the plaintexts m_0 and m_1 : the adversary has no advantage.

The proof follows by successively modifying the rules involved in the (perfect) simulation where the oracles \mathcal{P} and \mathcal{P}^{-1} are first simulated by using a perfectly random permutation P and its inverse P^{-1} . The last game provides a simulation of \mathcal{D}_{sk} , without inverting φ_{pk} .

Anyway, the simulation remains almost perfect unless the adversary asks the pre-image via φ_{pk} of the challenge ciphertext to the random permutation \mathcal{P}^{-1} : it thus helps to invert φ . The complete proof can be found in the full version of this paper [19].

4 The Random Oracle Model and OAEP

The above result is not so surprising, but the optimal bandwidth is a very good news. However the proof requires a full-domain random permutation, which is hard to find: practical block-ciphers have smaller block sizes. In this section, we present an instantiation of this random permutation, in the random oracle model only. The counter-part will be the need of a stronger assumption about the trapdoor one-way permutation: with a 3-round OAEP, a trapdoor partial-domain one-way permutation leads to an IND-CCA cryptosystem, without redundancy.

4.1 The 2-Round OAEP Case

Before studying the 3-round OAEP, let us first consider the more classical 2-round OAEP which can be described as follows: we use two hash functions \mathcal{G} and

\mathcal{H} before encrypting with a trapdoor one-way permutation φ_{pk} . More precisely, for encrypting a message m , one randomly chooses r , and computes s and t :

$$s = m \oplus \mathcal{G}(r) \quad t = r \oplus \mathcal{H}(s).$$

Then, the ciphertext is $c = \varphi_{\text{pk}}(s, t)$. For decryption, one computes

$$(s, t) = \psi_{\text{sk}}(c) \quad r = t \oplus \mathcal{H}(s) \quad m = s \oplus \mathcal{G}(r).$$

The usual way to prove the security of a scheme is to exploit an adversary to break the assumption (for instance, the partial-domain one-wayness of the permutation φ_{pk}). For that, we must simulate all the resources that the attacker can access, namely, the oracles \mathcal{G} , \mathcal{H} but also the decryption oracle \mathcal{D}_{sk} . For the above 2-round OAEP, the decryption oracle does not seem simulatable. The following attack game uses the same arguments as the counter-example shown by Shoup against the original OAEP security result [23]. Let us consider an attacker who chooses s, s' and calls for \mathcal{H} to get respectively $h = \mathcal{H}(s)$ and $h' = \mathcal{H}(s')$. Then it chooses t and computes $c = \varphi_{\text{pk}}(s, t)$. If it asks c to \mathcal{D}_{sk} , it gets the corresponding plaintext m . Then, it computes $t' = t \oplus h \oplus h'$ and $c' = \varphi_{\text{pk}}(s', t')$. If it asks c' to \mathcal{D}_{sk} , it gets the corresponding plaintext m' . One can easily see that, since $r' = r$, the relation $m \oplus m' = s \oplus s'$ should hold. But if the simulator can not detect that $r' = r$, it can not output a consistent value for m' .

Unfortunately, we did not find any easy way to make a consistent simulation for the 2-round OAEP. But a 3-round is more promising.

4.2 Description of the 3-Round OAEP

The public key is any trapdoor (partial-domain) one-way bijection φ_{pk} from a set E to a set F , while the private key is the inverse ψ_{sk} . For the sake of generality, we do not stick to binary sets (of the form $\{0, 1\}^k$): we just assume that there is an integer κ such that:

$$\{0\}^\kappa \times \{0, 1\}^{k+\ell} \subseteq E \subseteq \{0, 1\}^{\kappa+k+\ell} \quad (\text{identified to } \{0, 1\}^\kappa \times \{0, 1\}^k \times \{0, 1\}^\ell).$$

However, note that in the case that $E \neq 0^\kappa \parallel \{0, 1\}^{k+\ell}$ we won't get (as announced) a surjective encryption. But contrary to all the previous IND-CCA schemes, the proportion of valid ciphertexts (i.e., which are reachable) is greater than $1/2^\kappa$, which is not negligible: for efficient applications with RSA, it can be equal to $1/2$, or even 1 (by loosing a factor 2 in efficiency, one can get $\kappa = 0$, with the iterated-RSA [2]).

The encryption and decryption algorithms use three hash functions: \mathcal{F} , \mathcal{G} , \mathcal{H} (assumed to behave like random oracles in the security analysis):

$$\mathcal{F} : \{0, 1\}^k \rightarrow \{0, 1\}^\ell \quad \mathcal{G} : \{0, 1\}^\ell \rightarrow \{0, 1\}^k \quad \mathcal{H} : \{0, 1\}^{k+\kappa} \rightarrow \{0, 1\}^\ell.$$

Encryption Algorithm: The space of the plaintexts is $\mathcal{M} = \{0, 1\}^\ell$, the encryption algorithm uses random coins in $\mathcal{R} = \{0, 1\}^k$, and outputs a ciphertext c into F : on a plaintext $m \in \mathcal{M}$, and a random $r \in \mathcal{R}$, one computes

$$s = m \oplus \mathcal{F}(r) \quad t = r \oplus \mathcal{G}(s) \quad u = s \oplus \mathcal{H}(0^\kappa \parallel t) \quad c = \varphi_{\text{pk}}(0^\kappa, t, u).$$

Decryption Algorithm: On a ciphertext c , one first computes $(B, t, u) = \varphi_{\text{sk}}(c)$, where $B \in \{0, 1\}^\kappa$, $t \in \{0, 1\}^k$, $u \in \{0, 1\}^\ell$ and then

$$s = u \oplus \mathcal{H}(B \| t) \quad r = t \oplus \mathcal{G}(s) \quad m = s \oplus \mathcal{F}(r).$$

4.3 Security Result

About the 3-round OAEP, one can claim the following security result, which states that the IND-CCA security level is achieved under the (set) partial-domain one-wayness of the trapdoor permutation φ [15].

Theorem 2. *Let \mathcal{A} be any chosen-ciphertext adversary against the 3-round OAEP construction with the trapdoor permutation family φ , within time τ . After q_f , q_g , q_h and q_d queries to the random oracles \mathcal{F} , \mathcal{G} and \mathcal{H} , and the decryption oracle respectively,*

$$\begin{aligned} \text{Adv}_\pi^{\text{ind-cca}}(\tau) &\leq 2^\kappa \times \text{Succ}_\varphi^{\text{s-pd-ow}}(\tau + q_g \cdot q_h \times T_\varphi + q_d \cdot T_{lu}, q_h) \\ &\quad \frac{q_f}{2^k} + \frac{q_g}{2^\ell} + 2^\kappa \times \left(\frac{q_d(2q_g + q_d)}{2^\ell} + \frac{q_d(3q_f + 2q_d)}{2^k} \right) \end{aligned}$$

where T_φ is the time complexity for evaluating φ , and T_{lu} is the time complexity for a look up in a list.

Let us recall the definition of the (set) partial-domain one-wayness in our particular case, where \mathcal{A} is any algorithm which outputs a subset of $\{0, 1\}^k$ of size q :

$$\text{Succ}_\varphi^{\text{s-pd-ow}}(\mathcal{A}, q) = \Pr_{\substack{(B, t, u) \in E \\ \text{pk}}} [t \in \mathcal{A}(\varphi_{\text{pk}}(B, t, u))]$$

$$\text{and } \text{Succ}_\varphi^{\text{ow}}(\tau, q) = \max_{|\mathcal{A}| \leq \tau} \left\{ \text{Succ}_\varphi^{\text{s-pd-ow}}(\mathcal{A}, q) \right\},$$

is small for any reasonable time bound τ .

4.4 Sketch of the Proof

The goal of the proof is again to simulate the oracles. For simulating the random oracles, we use lists as usual to store the known answers. We simulate the decryption oracle as follows: when we receive a query y , either the corresponding s and t have both been asked to \mathcal{G} and \mathcal{H} , we can extract m , or one of them has not been asked, we can safely answer a random plaintext. However, such a plaintext-ciphertext relation implicitly defines several relations about the random oracles \mathcal{F} , \mathcal{G} and \mathcal{H} . We show that it is still possible to answer consistently. The challenge ciphertext also implicitly defines relations. We show that possible inconsistencies with the latter relations can not be detected by the adversary unless it has partially inverted the function φ_{pk} on the challenge ciphertext.

The proof is provided by a sequence of games, but for clarity reasons, we briefly explain only the distances between two consecutive games. The formal and full proofs are provided in the Appendix A.

Game \mathbf{G}_0 : The adversary is fed with the public key pk , and outputs a pair of messages (m_0, m_1) . Next a challenge ciphertext is produced by flipping a coin b and producing a ciphertext c^* of $m^* = m_b$. This ciphertext comes from a random $r^* \leftarrow \{0, 1\}^k$ and $c^* = \mathcal{E}(m_b, r^*) = \varphi_{\text{pk}}(0^\kappa, t, u)$. On input c^* , A_2 outputs bit b' in the time t . We denote by \mathbf{S}_0 the event $b' = b$ and use the same notation \mathbf{S}_n in any game \mathbf{G}_n below. Note that the adversary is given access to the decryption oracle \mathcal{D}_{sk} during both steps of the attack. The adversary can also ask the random oracles \mathcal{F} , \mathcal{G} , and \mathcal{H} .

Game \mathbf{G}_1 : The simulation in this game is presented on the Figure 1. We simulate the way that the challenge c^* is generated as the challenger would do, and we simulate the random oracles \mathcal{F} , \mathcal{G} , and \mathcal{H} , as well as the decryption oracle \mathcal{D}_{sk} , by maintaining lists $\mathcal{F}\text{-List}$, $\mathcal{G}\text{-List}$, $\mathcal{H}\text{-List}$ and $\mathcal{D}\text{-List}$ to deal with identical queries, since they all are deterministic. Since the simulation is perfect, we directly derive that

$$\Pr[\mathbf{S}_1] = \Pr[\mathbf{S}_0]. \quad (1)$$

Game \mathbf{G}_2 : We manufacture the challenge c^* independently of anything else.

► **Rule Chal⁽²⁾**

Choose randomly ahead of time $c^+ \xleftarrow{R} \mathcal{F}$ and set $c^* = c^+$.

Lemma 3. *Let us note (B^+, t^+, u^+) the pre-image of the challenge c^+ . We denote by AskH_2 the event that $B^+ || t^+$ has been asked to \mathcal{H} . Then,*

$$\Pr[\mathbf{S}_1] \leq \frac{1}{2} + \frac{q_f}{2^k} + \frac{q_g}{2^\ell} + 2^\kappa \times \Pr[\text{AskH}_2]. \quad (2)$$

Proof (Full proof in the Appendix A.1). The main idea in simulating this game is that we make the components of the challenge c^* (namely r^* , f^* , s^* , g^* , t^* , h^* , u^* and c^*) independent to m^* . We can do this by choosing ahead of time random values for r^* , s^* , and t^* , and we can see that a difference occurs when one of these values is asked to the corresponding oracle. On the other hand, when the challenge is independent to m^* , the attacker has only the chance of one half to guess the bit b . \square

Game \mathbf{G}_3 : In this game, we modify the simulation of the decryption oracle, by outputting a random message when the ciphertext has not been “correctly” encrypted. We thereafter define in a consistent way the values of the random oracles:

► **Rule Decrypt-TnoS⁽³⁾**

Choose $m \xleftarrow{R} \{0, 1\}^\ell$ and $g \xleftarrow{R} \{0, 1\}^k$,
then define $r = t \oplus g$ and $f = m \oplus s$.
Add (r, f) in $\mathcal{F}\text{-List}$, and (s, g) in $\mathcal{G}\text{-List}$.

► **Rule Decrypt-noT⁽³⁾**

Choose $m \xleftarrow{R} \{0, 1\}^\ell$, $h \xleftarrow{R} \{0, 1\}^\ell$ and $g \xleftarrow{R} \{0, 1\}^k$,
then define $s = u \oplus h$, $r = t \oplus g$ and $f = m \oplus s$.
Add (r, f) in $\mathcal{F}\text{-List}$, (s, g) in $\mathcal{G}\text{-List}$, and (B, t, h) in $\mathcal{H}\text{-List}$.

\mathcal{F}, \mathcal{G} and \mathcal{H} Oracles	<p>Query $\mathcal{F}(r)$: if a record (r, f) appears in \mathcal{F}-List, the answer is f. Otherwise the answer f is chosen randomly: $f \in \{0, 1\}^k$ and the record (r, f) is added in \mathcal{F}-List.</p> <hr/> <p>Query $\mathcal{G}(s)$: if a record (s, g) appears in \mathcal{G}-List, the answer is g. Otherwise the answer g is chosen randomly: $g \in \{0, 1\}^\ell$ and the record (s, g) is added in \mathcal{G}-List.</p> <p>► Rule EvalGAdd⁽¹⁾ Do nothing</p> <hr/> <p>Query $\mathcal{H}(B t)$: if a record (B, t, h) appears in \mathcal{H}-List, the answer is h. Otherwise the answer h is chosen randomly: $h \in \{0, 1\}^k$ and the record (B, t, h) is added in \mathcal{H}-List.</p>
\mathcal{D} Oracle	<p>Query $\mathcal{D}_{sk}(c)$: if a record (m, c) appears in \mathcal{D}-List, the answer is m. Otherwise the answer m is defined according to the following rules:</p> <p>► Rule Decrypt-Init⁽¹⁾ Compute $(B, t, u) = \psi_{sk}(c)$;</p> <p>Look up for $(B, t, h) \in \mathcal{H}$-List:</p> <ul style="list-style-type: none"> – if the record is found, compute $s = u \oplus h$. Look up for $(s, g) \in \mathcal{G}$-List: <ul style="list-style-type: none"> • if the record is found <p>► Rule Decrypt-TS⁽¹⁾ $h = \mathcal{H}(B t),$ $s = u \oplus h, \quad g = \mathcal{G}(s),$ $r = t \oplus g, \quad f = \mathcal{F}(r),$ $m = s \oplus f.$</p> • otherwise <p>► Rule Decrypt-TnoS⁽¹⁾ same as rule Decrypt-TS⁽¹⁾.</p> – otherwise <p>► Rule Decrypt-noT⁽¹⁾ same as rule Decrypt-TS⁽¹⁾.</p> <p>Answer m and add (m, c) to \mathcal{D}-List.</p>
Challenger	<p>For two messages (m_0, m_1), flip a coin b and set $m^* = m_b$, choose randomly r^*, then answer c^*, where</p> <p>► Rule Chal⁽¹⁾ $f^* = \mathcal{F}(r^*), \quad s^* = m^* \oplus f^*,$ $g^* = \mathcal{G}(s^*), \quad t^* = r^* \oplus g^*,$ $h^* = \mathcal{H}(0^\kappa t^*), \quad u^* = s^* \oplus h^*.$ Compute $c^* = \varphi_{pk}(0^\kappa, t^*, u^*)$.</p>

Fig. 1. Formal Simulation of the IND-CCA Game against 3-OAEP.

Lemma 4.

$$|\Pr[\text{AskH}_3] - \Pr[\text{AskH}_2]| \leq \frac{q_d(q_g + q_d)}{2^\ell} + 2 \frac{q_d(q_f + q_d)}{2^k}. \quad (3)$$

Proof (Full proof in the Appendix A.2). In the proof, one successively modifies the simulation of the decryption oracle, just changing the order of elements to be randomly chosen, so that the decryption of a ciphertext which has not been correctly encrypted is a truly random plaintext. \square

Game \mathbf{G}_4 : In this game, we delay the explicit definitions of some oracle answers implicitly defined by some plaintext-ciphertext relations: we do not introduce them during the simulation of the decryption oracle, but when s is asked to \mathcal{G} . Some problems may appear if the implicitly defined answers are asked before $\mathcal{G}(s)$ is queried.

► **Rule Decrypt-TnoS⁽⁴⁾**

| Choose $m \xleftarrow{R} \{0, 1\}^\ell$.

► **Rule Decrypt-noT⁽⁴⁾**

| Choose $m \xleftarrow{R} \{0, 1\}^\ell$.

► **Rule EvalGAdd⁽⁴⁾**

| Look up for $(B, t, h) \in \mathcal{H}\text{-List}$ and $(m, c) \in \mathcal{D}\text{-List}$ such that $c = \varphi_{\text{pk}}(B, t, h \oplus s)$.
 If the record is found, we compute $r = t \oplus g$ and $f = m \oplus s$, and finally add (r, f) in $\mathcal{F}\text{-List}$.

Lemma 5.

$$|\Pr[\text{AskH}_4] - \Pr[\text{AskH}_3]| \leq \frac{q_d \cdot q_f}{2^k} + \frac{q_d \cdot q_g}{2^\ell}. \quad (4)$$

Proof (Full proof in the Appendix A.3). Since we don't store anymore (r, f) , (s, g) , (B, t, h) , inconsistencies could occur when $B||t$, s or r are asked. For solving this problem, we modify the rule **EvalGAdd** by defining in a consistent way $\mathcal{F}(r)$ at the moment that s is asked to \mathcal{G} . But there is still a problem if r is asked before $\mathcal{G}(s)$ is queried, or if s is asked before $\mathcal{H}(B||t)$ is queried. \square

Game \mathbf{G}_5 : We now complete the simulation of the oracle \mathcal{D}_{sk} . We don't ask any query to ψ_{sk} . Intuitively, if both $B||t$ and s have been asked, we can easily find them, and then m . Otherwise, we give a random answer as in the game \mathbf{G}_4 .

► **Rule Decrypt-Init⁽⁵⁾**

| Look up for $(B, t, h) \in \mathcal{H}\text{-List}$ and $(s, g) \in \mathcal{G}\text{-List}$ such that $\varphi_{\text{pk}}(B, t, s \oplus h) = c$.
 – if the record is found, we furthermore define $u = s \oplus h$.
 – otherwise, we take $B = \perp, t = \perp, u = \perp$.

► **Rule Decrypt-TS⁽⁵⁾**

$$\mid \quad r = t \oplus g, \quad f = \mathcal{F}(r), \quad m = s \oplus f.$$

The two games \mathbf{G}_5 and \mathbf{G}_4 are perfectly indistinguishable. In fact, in the first case, nothing is modified and in the second case, by making $B = \perp, t = \perp, u = \perp$, the answer of the decryption oracle for the question c will be a random m as in the game \mathbf{G}_4 :

$$\Pr[\text{AskH}_5] = \Pr[\text{AskH}_4]. \quad (5)$$

Simply outputting the list of queries to \mathcal{H} during this game, one gets:

$$\Pr[\text{AskH}_5] \leq \text{Succ}_{\varphi}^{\text{s-pd-ow}}(\tau', q_h), \quad (6)$$

where τ' is the running time of the simulation in this game: $\tau' \leq q_g \cdot q_h \times T_{\varphi} + q_d \times T_{lu}$. We can indeed perform the simulation granted an additional list $\mathcal{GH}\text{-List}$ which contains all the tuples (B, t, h, s, g, y) where $(B, t, h) \in \mathcal{H}\text{-List}$, $(s, g) \in \mathcal{G}\text{-List}$ and $y = \varphi_{pk}(B, t, s \oplus h)$. This concludes the proof of the Theorem.

4.5 Special Cases

In the particular but classical case where $\kappa = 0$ and $k \leq \ell$, one can claim

Theorem 6. *Let \mathcal{A} be any chosen-ciphertext adversary against the 3-round OAEP construction with the trapdoor permutation family φ , within time τ . After q_o and q_d queries to the random oracles and the decryption oracle respectively,*

$$\text{Adv}_{\pi}^{\text{ind-cca}}(\tau) \leq \text{Succ}_{\varphi}^{\text{s-pd-ow}}(\tau + q_o^2 \times T_{\varphi} + q_d \times T_{lu}, q_o) + \frac{2q_o + q_d(5q_o + 2q_d)}{2^k}$$

where T_{φ} is the time complexity for evaluating φ , and T_{lu} is the time complexity for a look up in a list.

5 Conclusion

We have described the Full-Domain Permutation encryption which is IND-CCA without redundancy and provides an optimal bandwidth. In the random oracle model, we have shown that the absence redundancy can be obtained by considering the 3-round OAEP construction. However, the bandwidth is not optimal, and the security relies on the strong partial-domain one-wayness assumption.

Acknowledgments

We thank Anand Desai for fruitful discussions.

References

1. M. Abdalla, M. Bellare, and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In *CT – RSA ’01*, LNCS 2020, pages 143–158. Springer-Verlag, Berlin, 2001.
2. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-Privacy in Public-Key Encryption. In *Asiacrypt ’01*, LNCS 2248, pages 566–582. Springer-Verlag, Berlin, 2001.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto ’98*, LNCS 1462, pages 26–45. Springer-Verlag, Berlin, 1998.
4. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.
5. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt ’94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.
6. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt ’96*, LNCS 1070, pages 399–416. Springer-Verlag, Berlin, 1996.
7. M. Bellare and A. Sahai. Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *Crypto ’99*, LNCS 1666, pages 519–536. Springer-Verlag, Berlin, 1999.
8. D. Boneh. Simplified OAEP for the RSA and Rabin Functions. In *Crypto ’01*, LNCS 2139, pages 275–291. Springer-Verlag, Berlin, 2001.
9. J.-S. Coron. On the Exact Security of Full-Domain-Hash. In *Crypto ’00*, LNCS 1880, pages 229–235. Springer-Verlag, Berlin, 2000.
10. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *Crypto ’98*, LNCS 1462, pages 13–25. Springer-Verlag, Berlin, 1998.
11. A. Desai. New Paradigms for Constructing Symmetric Encryption Schemes Secure Against Chosen-Ciphertext Attack. In *Crypto ’00*, LNCS 1880, pages 394–412. Springer-Verlag, Berlin, 2000.
12. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. In *Proc. of the 23rd STOC*. ACM Press, New York, 1991.
13. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In *PKC ’99*, LNCS 1560, pages 53–68. Springer-Verlag, Berlin, 1999.
14. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Crypto ’99*, LNCS 1666, pages 537–554. Springer-Verlag, Berlin, 1999.
15. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is Secure under the RSA Assumption. *Journal of Cryptology*, 2003. To appear.
16. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
17. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proc. of the 22nd STOC*, pages 427–437. ACM Press, New York, 1990.
18. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT – RSA ’01*, LNCS 2020, pages 159–175. Springer-Verlag, Berlin, 2001.

19. D. H. Phan and D. Pointcheval. Chosen-Ciphertext Security without Redundancy. In *Asiacrypt '03*, LNCS. Springer-Verlag, Berlin, 2003. Full version available at <http://www.di.ens.fr/users/pointche>
20. D. Pointcheval. Chosen-Ciphertext Security for any One-Way Cryptosystem. In *PKC '00*, LNCS 1751, pages 129–146. Springer-Verlag, Berlin, 2000.
21. C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto '91*, LNCS 576, pages 433–444. Springer-Verlag, Berlin, 1992.
22. R. Rivest, A. Shamir, and A. Tauman, How to Leak a Secret. In *Asiacrypt '01*, LNCS 2248, pages 552–565. Springer-Verlag, Berlin, 2001.
23. V. Shoup. OAEP Reconsidered. In *Crypto '01*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001.

A Complements for the Proof of the Theorem 2

A.1 Proof of Lemma 3

Game $\mathbf{G}_{1.1}$: For proving this lemma, we present a more detailed sequence of games from the game \mathbf{G}_1 to the game $\mathbf{G}_{1.2}$. We first make the value of the random seed r^* explicit and move its generation up-front.

► **Rule Chal^(1.1)**

The two values $r^+ \xleftarrow{R} \{0,1\}^k$, $f^+ \xleftarrow{R} \{0,1\}^\ell$ have been chosen ahead of time, then

$$r^* = r^+, \quad f^* = f^+, \quad s^* = m^* \oplus f^+, \quad g^* = \mathcal{G}(s^*),$$

$$t^* = r^+ \oplus g^*, \quad h^* = \mathcal{H}(0^\kappa \| t^*), \quad u^* = s^* \oplus h^*.$$

Compute $c^* = \varphi_{\text{pk}}(0^\kappa, t^*, u^*)$.

The two games $\mathbf{G}_{1.1}$ and \mathbf{G}_1 are perfectly indistinguishable unless r^* has been asked for \mathcal{F} . We define this event $\text{AskF}_{1.1}$. We have:

$$|\Pr[\text{S}_{1.1}] - \Pr[\text{S}_1]| \leq \Pr[\text{AskF}_{1.1}]. \quad (7)$$

In this game, f^+ is used in (s, t) but does not appear in the computation since $\mathcal{F}(r^+)$ is not defined to be equal to f^+ . Thus, the input to \mathcal{A}_2 follows a distribution that does not depend on b . Accordingly:

$$\Pr[\text{S}_{1.1}] = \frac{1}{2}. \quad (8)$$

Game $\mathbf{G}_{1.2}$: In this game, instead of defining s^* from f^* which is a random value f^+ , we randomly choose s^* and then we define f^+ from s^* . Because s^* is chosen randomly, we give a random answer for the question s^* to \mathcal{G} .

►Rule Chal^(1.2)

The values $r^+ \xleftarrow{R} \{0, 1\}^k$, $s^+ \xleftarrow{R} \{0, 1\}^\ell$, $g^+ \xleftarrow{R} \{0, 1\}^k$ have been chosen ahead of time, then

$$r^* = r^+ \quad s^* = s^+ \quad g^* = g^+ \quad f^* = s^+ \oplus m^*$$

$$t^* = r^+ \oplus g^+ \quad h^* = \mathcal{H}(t^*) \quad u^* = s^+ \oplus h^*.$$

Compute $c^* = \varphi_{\text{pk}}(0^\kappa, t^*, u^*)$.

The two games $\mathbf{G}_{1.2}$ and $\mathbf{G}_{1.1}$ are perfectly indistinguishable unless s^* is asked for \mathcal{G} . We define this event $\text{AskG}_{1.2}$. We have:

$$|\Pr[\text{AskF}_{1.2}] - \Pr[\text{AskF}_{1.1}]| \leq \Pr[\text{AskG}_{1.2}]. \quad (9)$$

In this game, $r^+ = t^* \oplus g^+$ is uniformly distributed, and independently of the adversary's view since g^+ is never revealed:

$$\Pr[\text{AskF}_{1.2}] = \frac{qf}{2^k}. \quad (10)$$

Game $\mathbf{G}_{1.3}$: Similarly to the above game, instead of defining t^* from a random g^+ , we randomly choose t^* and then we define g^+ from t^* . Because t^* is chosen randomly, we give a random answer for the question $(0^\kappa || t^*)$ to \mathcal{H} .

►Rule Chal^(1.3)

The values $r^+ \xleftarrow{R} \{0, 1\}^k$, $s^+ \xleftarrow{R} \{0, 1\}^\ell$, $t^+ \xleftarrow{R} \{0, 1\}^k$, $h^+ \xleftarrow{R} \{0, 1\}^\ell$ have been chosen ahead of time, then

$$r^* = r^+ \quad s^* = s^+ \quad t^* = t^+ \quad h^* = h^+$$

$$f^* = s^+ \oplus m^* \quad g^* = t^+ \oplus r^+ \quad u^* = s^+ \oplus h^+.$$

Compute $c^* = \varphi_{\text{pk}}(0^\kappa, t^*, u^*)$.

The two games $\mathbf{G}_{1.3}$ and $\mathbf{G}_{1.2}$ are perfectly indistinguishable unless $0^\kappa || t^*$ is asked for \mathcal{H} . We define this event $\text{AskH}_{1.3}$. We have:

$$|\Pr[\text{AskG}_{1.3}] - \Pr[\text{AskG}_{1.2}]| \leq \Pr[\text{AskH}_{1.3}]. \quad (11)$$

In this game, $s^+ = u^* \oplus h^+$ is uniformly distributed, and independently of the adversary's view since h^+ is never revealed:

$$\Pr[\text{AskG}_{1.3}] = \frac{qg}{2^\ell}.$$

Game $\mathbf{G}_{1.4}$: We manufacture the challenge c^* independently of anything else.

►Rule Chal^(1.4)

The values $t^+ \xleftarrow{R} \{0, 1\}^k$, $u^+ \xleftarrow{R} \{0, 1\}^\ell$ have been chosen ahead of time.

Compute $c^* = \varphi_{\text{pk}}(0^\kappa, t^+, u^+)$.

The distribution of c^* remains the same:

$$\Pr[\text{AskH}_{1.4}] = \Pr[\text{AskH}_{1.3}]. \quad (12)$$

Game $\mathbf{G}_{1.5}$: We choose the challenge c^* uniformly in the space \mathcal{F} .

► **Rule $\text{ChalC}^{(1.5)}$**

The value $c^+ \xleftarrow{R} \mathcal{F}$ is chosen randomly ahead of time, then $c^* = c^+$.

We can write c^+ as $\varphi_{\text{pk}}(B^+, t^+, h^+)$. We define $\text{AskH}_{1.5}$ the event that $B^+ || t^+$ is asked to \mathcal{H} . In the case $B^+ = 0^\kappa$, which event is denoted by GoodB and which probability is at least $1/2^\kappa$, this game is identical to the previous one:

$$\begin{aligned} \Pr[\text{AskH}_{1.5}] &= \Pr[\text{AskH}_{1.5} \wedge \text{GoodB}] + \Pr[\text{AskH}_{1.5} \wedge \neg \text{GoodB}] \\ &\geq \Pr[\text{AskH}_{1.5} | \text{GoodB}] \cdot \Pr[\text{GoodB}] \geq \Pr[\text{AskH}_{1.4}] \cdot \frac{1}{2^\kappa}. \end{aligned} \quad (13)$$

To conclude the proof of the lemma, one first notes that the games $\mathbf{G}_{1.5}$ and \mathbf{G}_2 are identical, and thus $\Pr[\text{AskH}_{1.5}] = \Pr[\text{AskH}_2]$. Then, combining all the above equations, on gets

$$\begin{aligned} \Pr[S_1] &\leq \Pr[S_{1.1}] + \Pr[\text{AskF}_{1.1}] \leq \frac{1}{2} + \Pr[\text{AskF}_{1.1}] \\ &\leq \frac{1}{2} + \Pr[\text{AskF}_{1.2}] + \Pr[\text{AskG}_{1.2}] \\ &\leq \frac{1}{2} + \Pr[\text{AskF}_{1.2}] + \Pr[\text{AskG}_{1.3}] + 2^\kappa \cdot \Pr[\text{AskH}_{1.5}] \\ &\leq \frac{1}{2} + \frac{q_f}{2^k} + \frac{q_g}{2^\ell} + 2^\kappa \cdot \Pr[\text{AskH}_2]. \end{aligned}$$

A.2 Proof of Lemma 4

Game $\mathbf{G}_{2.1}$: First, we modify the rule **Decrypt-noT** by not calling anymore the oracles \mathcal{G} and \mathcal{H} . Let us remind that the adversary asks a \mathcal{D} -query on $c = \varphi_{\text{pk}}(B, t, u)$ such that $\mathcal{H}(B || t)$ has never been queried.

► **Rule $\text{Decrypt-noT}^{(2.1)}$**

Choose $h \xleftarrow{R} \{0, 1\}^\ell$ and set $s = u \oplus h$.
 Choose $g \xleftarrow{R} \{0, 1\}^k$ and set $r = t \oplus g$.
 Compute $f = \mathcal{F}(r)$ and set $m = s \oplus f$.
 Add (s, g) in $\mathcal{G}\text{-List}$, (B, t, h) in $\mathcal{H}\text{-List}$.

The two games $\mathbf{G}_{2.1}$ and \mathbf{G}_2 are perfectly indistinguishable unless s is already in $\mathcal{G}\text{-List}$. Because $B || t$ has not been queried to \mathcal{H} , $h = \mathcal{H}(B || t)$ is uniformly

distributed and therefore, we can consider s as a uniform variable. So, the probability that s has already been queried to \mathcal{G} is $(q_g + q_d)/2^\ell$:

$$|\Pr[\text{AskH}_{2.1}] - \Pr[\text{AskH}_2]| \leq q_d(q_g + q_d)/2^\ell. \quad (14)$$

Game $\mathbf{G}_{2.2}$: In this game, we modify again the rule $\text{Decrypt-noT}^{(2.2)}$ by not querying the oracle \mathcal{F} either:

► **Rule $\text{Decrypt-noT}^{(2.2)}$**

Choose $h \xleftarrow{R} \{0, 1\}^\ell$ and set $s = u \oplus h$. Choose $g \xleftarrow{R} \{0, 1\}^k$ and set $r = t \oplus g$. Choose $f \xleftarrow{R} \{0, 1\}^\ell$ and set $m = s \oplus f$. Add (r, f) in $\mathcal{F}\text{-List}$, (s, g) in $\mathcal{G}\text{-List}$, (B, t, h) in $\mathcal{H}\text{-List}$.

The two games $\mathbf{G}_{2.2}$ and $\mathbf{G}_{2.1}$ are perfectly indistinguishable unless r is already in $\mathcal{F}\text{-List}$. Since g is randomly chosen, we can consider r as a uniform variable. So, the probability that r has already been queried to \mathcal{F} is less than $(q_f + q_d)/2^k$:

$$|\Pr[\text{AskH}_{2.2}] - \Pr[\text{AskH}_{2.1}]| \leq q_d(q_f + q_d)/2^k. \quad (15)$$

Game $\mathbf{G}_{2.3}$: Still about the rule Decrypt-noT , instead of defining m from a random f , we first choose m and then we define f from m :

► **Rule $\text{Decrypt-noT}^{(2.3)}$**

Choose $m \xleftarrow{R} \{0, 1\}^\ell$. Choose $h \xleftarrow{R} \{0, 1\}^\ell$ and set $s = u \oplus h$. Choose $g \xleftarrow{R} \{0, 1\}^k$ and set $r = t \oplus g$. Compute $f = m \oplus s$. Add (r, f) in $\mathcal{F}\text{-List}$, (s, g) in $\mathcal{G}\text{-List}$, (B, t, h) in $\mathcal{H}\text{-List}$.
--

The two games $\mathbf{G}_{2.3}$ and $\mathbf{G}_{2.2}$ are perfectly indistinguishable:

$$\Pr[\text{AskH}_{2.3}] = \Pr[\text{AskH}_{2.2}]. \quad (16)$$

Game $\mathbf{G}_{2.4}$: We now modify the rule Decrypt-TnoS by not calling anymore the oracles \mathcal{F} and \mathcal{G} . About this rule, the adversary asks for the decryption of $c = \varphi_{\text{pk}}(B, t, u)$ such that $h = \mathcal{H}(B||t)$ is known, but $s = u \oplus h$ has never been queried to \mathcal{G} .

► **Rule $\text{Decrypt-TnoS}^{(2.4)}$**

Choose $g \xleftarrow{R} \{0, 1\}^k$ and set $r = t \oplus g$. Choose $f \xleftarrow{R} \{0, 1\}^\ell$ and set $m = s \oplus f$. Add (r, f) in $\mathcal{F}\text{-List}$, (s, g) in $\mathcal{G}\text{-List}$.
--

The two games $\mathbf{G}_{2.4}$ and $\mathbf{G}_{2.3}$ are perfectly indistinguishable unless r is already in $\mathcal{F}\text{-List}$. Since g is randomly chosen (s is not in $\mathcal{G}\text{-List}$), we can consider r

as a uniform variable. So, the probability that r is queried to \mathcal{F} is less than $(q_f + q_d)/2^k$:

$$|\Pr[\text{AskH}_{2.4}] - \Pr[\text{AskH}_{2.3}]| \leq q_d(q_f + q_d)/2^k. \quad (17)$$

Game $\mathbf{G}_{2.5}$: As above, in the rule **Decrypt-TnoS**, instead of defining m from a random f , we first choose m and then we define f from m :

► **Rule Decrypt-TnoS^(2.5)**

Choose $m \xleftarrow{R} \{0, 1\}^\ell$.
 Choose $g \xleftarrow{R} \{0, 1\}^k$ and set $r = t \oplus g$.
 Compute $f = m \oplus s$.
 Add (r, f) in \mathcal{F} -List, (s, g) in \mathcal{G} -List.

The two games $\mathbf{G}_{2.5}$ and $\mathbf{G}_{2.4}$ are perfectly indistinguishable:

$$\Pr[\text{AskH}_{2.5}] = \Pr[\text{AskH}_{2.4}]. \quad (18)$$

A.3 Proof of Lemma 5

Game $\mathbf{G}_{3.1}$: In this game, we don't store anymore (s, g) in \mathcal{G} -List, nor (r, f) in \mathcal{F} -List and we modify the simulation of \mathcal{G} , so that \mathcal{F} -List is built as soon as possible:

► **Rule Decrypt-TnoS^(3.1)**

Choose $m \xleftarrow{R} \{0, 1\}^\ell$.

► **Rule Decrypt-noT^(3.1)**

Choose $h \xleftarrow{R} \{0, 1\}^\ell$.
 Choose $m \xleftarrow{R} \{0, 1\}^\ell$.
 Add (B, t, h) in \mathcal{H} -List.

► **Rule EvalGAdd^(3.1)**

Search $(B, t, h) \in \mathcal{H}$ -List and $(m, c) \in \mathcal{D}$ -List such that $c = \varphi_{\text{pk}}(B, t, h \oplus s)$. If the record is found, we compute $r = t \oplus g$, $f = m \oplus s$ and add (r, f) in \mathcal{F} -List.

The two games $\mathbf{G}_{3.1}$ and \mathbf{G}_3 are perfectly indistinguishable unless r is asked to \mathcal{F} before s is asked to \mathcal{G} , we denote this event by **AskRbS**. In fact, if r is asked after s , at the moment that s is asked, by the above simulation of \mathcal{G} , we will find out (B, t, h) and therefore (r, f) is computed and added in \mathcal{F} -List as in the game \mathbf{G}_3 .

$$|\Pr[\text{AskH}_{3.1}] - \Pr[\text{AskH}_3]| \leq \Pr[\text{AskRbS}_{3.1}]. \quad (19)$$

Until s is asked, g is a uniform variable, so is r . Therefore, the probability that r has been asked to \mathcal{F} is $q_f/2^k$:

$$\Pr[\text{AskRbS}_{3.1}] \leq q_d \cdot q_f/2^k. \quad (20)$$

Game $\mathbf{G}_{3.2}$: We continue to simulate the oracle \mathcal{D}_{sk} . We use the following rule:

► **Rule Decrypt-noT^(3.2)**
 | Choose $m \xleftarrow{R} \{0, 1\}^\ell$.

In this game, we don't store anymore (B, t, h) in $\mathcal{H}\text{-List}$. In the $\mathbf{G}_{3.1}$, for the question t , we answer randomly h , so the attacker in the two games $\mathbf{G}_{3.2}$ and $\mathbf{G}_{3.1}$ can not distinguish the answers of a question to \mathcal{H} . Nevertheless, $\mathcal{H}\text{-List}$ has been changed and therefore, the answer for a question to \mathcal{F} can be changed. We easily see that the two games $\mathbf{G}_{3.2}$ and $\mathbf{G}_{3.1}$ are perfectly indistinguishable unless s is asked to \mathcal{G} before $B||t$ is asked to \mathcal{H} , we denote this event by AskSbT . In fact, if s is asked to \mathcal{G} after $B||t$ is asked to \mathcal{H} , at the moment s is asked, by the above simulation of \mathcal{G} , we will find out (B, t, h) and therefore (r, f) is computed and added in $\mathcal{F}\text{-List}$ as in the game $\mathbf{G}_{3.1}$.

$$|\Pr[\text{AskH}_{3.2}] - \Pr[\text{AskH}_{3.1}]| \leq \Pr[\text{AskSbT}_{3.2}]. \quad (21)$$

Until $B||t$ is asked to \mathcal{H} , h is a uniform variable, so is $s = u \oplus h$. Therefore, the probability that s has been asked to \mathcal{G} is $q_g/2^\ell$:

$$\Pr[\text{AskSbT}_{3.2}] \leq q_d \cdot q_g/2^\ell.$$

Some RSA-Based Encryption Schemes with Tight Security Reduction

Kaoru Kurosawa¹ and Tsuyoshi Takagi²

¹ Ibaraki University

4-12-1 Nakanarusawa, Hitachi, Ibaraki, 316-8511, Japan

`kurosawa@cis.ibaraki.ac.jp`

² Technische Universität Darmstadt

Fachbereich Informatik

Alexanderstr.10, D-64283 Darmstadt, Germany

`ttakagi@cdc.informatik.tu-darmstadt.de`

Abstract. In this paper, we study some RSA-based semantically secure encryption schemes (IND-CPA) in the standard model. We first derive the exactly tight one-wayness of Rabin-Paillier encryption scheme which assumes that factoring Blum integers is hard. We next propose the first IND-CPA scheme whose one-wayness is equivalent to factoring *general* $n = pq$ (not factoring Blum integers). Our reductions of one-wayness are very tight because they require only one decryption-oracle query.

Keywords: Factoring, semantic security, tight reduction, RSA-Paillier, Rabin-Paillier.

1 Introduction

1.1 Background

An encryption scheme should have strong one-wayness as well as high semantic security. Therefore, it is desirable to construct a semantically secure encryption scheme whose one-wayness is equivalent to factoring $n = pq$ in the *standard* model. (There are several provably secure constructions in the *random oracle* model. For example, see [Sho01,FOPS01,Bon01].)

RSA-Paillier encryption scheme is semantically secure against chosen plaintext attacks (IND-CPA) in the standard model under the RSA-Paillier assumption [CGHN01]. The assumption claims that

$$SMALL_{RSAP} = \{r^e \bmod n^2 | r \in Z_n\} \text{ and}$$

$$LARGE_{RSAP} = \{r^e \bmod n^2 | r \in Z_{n^2}\}$$

are indistinguishable, where (n, e) is the public-key of RSA. Further, it is one-way if breaking RSA is hard. The latter problem was first raised by [ST02] and finally proved by [CNS02] using LLL algorithm of lattice theory.

On the other hand, $n(=pq)$ is called a Blum integer if $p = q = 3 \bmod 4$. Galindo et al. recently considered Rabin-Paillier encryption scheme and showed that it is one-way if factoring Blum integers is hard [GMMV03].

However, there is a large gap between the one-wayness which they proved and the difficulty of factoring. That is, suppose that the one-wayness is broken with probability ε . Then what Galindo et al. proved is that Blum integers can be factored with probability ε^2 . Further the factoring problem is restricted to *Blum* integers, but not *general* p, q .

(The one-wayness of Okamoto-Uchiyama scheme [OU98] is equivalent to factoring $n = p^2q$, but not $n = pq$.)

1.2 Our Contribution

In this paper, we study the tight one-wayness of some RSA-based semantically secure encryption schemes (IND-CPA) in the standard model, where the one-wayness must be equivalent to factoring $n = pq$.

We first show that Rabin-Paillier encryption scheme has no gap between the *real* one-wayness and the difficulty of factoring Blum integers. (In other words, we give a factoring algorithm with success probability ε .) Our proof technique is quite different from previous proofs. In particular:

- Our proof technique requires only *one* decryption-oracle query while the previous proofs for RSA/Rabin-Paillier encryption schemes require *two* oracle queries [CNS02,GMMV03].
- No LLL algorithm is required, which was essentially used in the previous proofs for RSA/Rabin-Paillier schemes [CNS02,GMMV03].

We next propose the first IND-CPA scheme such that the one-wayness is equivalent to factoring *general* $n = pq$ (not factoring *Blum* integers). The one-wayness is proved by applying our proof technique as mentioned above. Therefore, our security reduction of one-wayness is very tight. That is, there is almost no gap between the one-wayness and the hardness of the general factoring problem.

The proposed scheme is obtained from an encryption scheme presented by Kurosawa et al. [KIT88,KOMM01]. The semantic security holds under a natural extension of RSA-Paillier assumption. That is, it is semantically secure (IND-CPA) if two distributions $SMALL_{RSAK}$ and $LARGE_{RSAK}$ are indistinguishable, where we define $SMALL_{RSAK}$ and $LARGE_{RSAK}$ as appropriate subsets of $SMALL_{RSAP}$ and $LARGE_{RSAP}$, respectively. We also show a close relationship between our assumption and RSA-Paillier assumption.

This paper is organized as follows: In Section 2, we describe notions required for the security description in this paper. In Section 3, the exact security reduction algorithm for Rabin-Paillier encryption scheme is presented. In Section 4, the proposed scheme is presented. In Section 5, we prove that the one-wayness of the proposed scheme is as hard as general factoring problem. In Section 6, we discuss the semantic security of the proposed scheme. Sec.7 includes some final comments.

Related works: Cramer and Shoup showed an semantically secure encryption scheme against chosen ciphertext attacks (IND-CCA) under the decision Diffie-

Hellamn assumption [CS98]. They recently showed a general framework to construct IND-CCA schemes [CS02].

It will be a further work to develop an IND-CCA scheme whose one-wayness is equivalent to the factoring problem in the standard model. We hope that our results provide us a good starting point to this challenging problem.

2 Security of Encryption Schemes

PPT will denote a “probabilistic polynomial time”.

2.1 Encryption Scheme

A public-key encryption scheme $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms. The key generation algorithm \mathcal{K} outputs (pk, sk) on input 1^l , where pk is a public key, sk is the secret key and l is a security parameter. We write $(pk, sk) \xleftarrow{R} \mathcal{K}$. The encryption algorithm \mathcal{E} outputs a ciphertext c on input the public key pk and a plaintext (message) m ; we write $c \xleftarrow{R} \mathcal{E}_{pk}(m)$. The decryption algorithm \mathcal{D} outputs m or *reject* on input the secret key sk and a ciphertext c ; we write $x \leftarrow \mathcal{D}_{sk}(c)$, where $x = m$ or *reject*. We require that $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m$ for each plaintext m . \mathcal{K} and \mathcal{E} are PPT algorithms, and \mathcal{D} is a polynomial time algorithm.

2.2 One-Wayness

The one-wayness problem is as follows: given a public key pk and a ciphertext c , find the plaintext m such that $c \xleftarrow{R} \mathcal{E}_{pk}(m)$. Formally, for an adversary A , consider an experiment as follows.

$$(pk, sk) \xleftarrow{R} \mathcal{K}, c \xleftarrow{R} \mathcal{E}_{pk}(m), \tilde{m} \xleftarrow{R} A(pk, c).$$

where m is randomly chosen from the domain of pk . Let

$$Adv_{\mathcal{PE}}^{ow}(A) = \Pr(\tilde{m} = m).$$

For any $t > 0$, define

$$Adv_{\mathcal{PE}}^{ow}(t) = \max_A Adv_{\mathcal{PE}}^{ow}(A),$$

where the maximum is over all A who run in time t .

Definition 1. We say that \mathcal{PE} is (t, ε) -one-way if $Adv_{\mathcal{PE}}^{ow}(t) < \varepsilon$. We also say that \mathcal{PE} is one-way if $Adv_{\mathcal{PE}}^{ow}(A)$ is negligible for any PPT adversary A .

2.3 Semantic Security

We say that a public-key encryption scheme $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is semantically secure against chosen plaintext attacks (SS-CPA) if it is hard to find any (partial) information on m from c . This notion is equivalent to indistinguishability (IND-CPA), which is described as follows [BDPR98, Gol01].

We consider an adversary $B = (B_1, B_2)$ as follows. In the “find” stage, B_1 takes a public key pk and outputs $(m_0, m_1, state)$, where m_0 and m_1 are two equal length plaintexts and $state$ is some state information. In the “guess” stage, B_2 gets a challenge ciphertext $c \xleftarrow{R} \mathcal{E}_{pk}(m_b)$ from an oracle, where b is a randomly chosen bit. B_2 finally outputs a bit \tilde{b} . We say that an encryption scheme \mathcal{PE} is secure in the sense of IND-CPA if $|\Pr(\tilde{b} = b) - 1/2|$ is negligible.

Formally, for each security parameter l , let

$$(pk, sk) \xleftarrow{R} \mathcal{K}, (m_0, m_1, state) \xleftarrow{R} B_1(pk), c \xleftarrow{R} \mathcal{E}_{pk}(m_b), \tilde{b} \xleftarrow{R} B_2(c, state).$$

Definition 2. We say that \mathcal{PE} is secure in the sense of indistinguishability against chosen-plaintext attack (IND-CPA) if

$$\text{Adv}_{\mathcal{PE}}^{\text{ind}}(B) \triangleq |\Pr(\tilde{b} = b) - 1/2|$$

is negligible for any PPT adversary B .

If an adversary $B = (B_1, B_2)$ is allowed to access the decryption oracle $\mathcal{D}_{sk}(\cdot)$, we denote it by $B^{\mathcal{D}} = (B_1^{\mathcal{D}}, B_2^{\mathcal{D}})$. If $\text{Adv}_{\mathcal{PE}}^{\text{ind}}(B^{\mathcal{D}})$ is negligible for any PPT adversary $B^{\mathcal{D}}$, we say that \mathcal{PE} is secure in the sense of indistinguishability against adaptive chosen-ciphertext attack (IND-CCA).

2.4 Factoring Assumptions

The general factoring problem is to factor $n = pq$, where p and q are two primes such that $|p| = |q|$. Formally, for an factoring algorithm B , consider the following experiment. Generate two primes p and q such that $|p| = |q|$ randomly. Give $n = pq$ to B . We say that B succeeds if B can output p or q .

Definition 3. We say that the general factoring problem is (t, ε) -hard if $\Pr(B \text{ succeeds}) < \varepsilon$ for any B who runs in time t . We also say that it is hard if $\Pr(B \text{ succeeds})$ is negligible for any PPT algorithm B .

The general factoring assumption claims that the general factoring problem is hard.

We say that $n(= pq)$ is a *Blum* integer if p and q are prime numbers such that $p = q = 3 \pmod{4}$ and $|p| = |q|$. The *Blum*-factoring problem is defined similarly. *Blum*-factoring assumption claims that the *Blum*-factoring problem is hard.

3 Exact One-Wayness of Rabin-Paillier Scheme

Galindo et al. recently constructed Rabin-Paillier encryption scheme [GMMV03] and showed that its one-wayness is as hard as factoring Blum integers, where $n = pq$ is called a Blum integer if $p = q = 3 \pmod{4}$. However, there is a polynomially bounded gap between the difficulty of factoring and the *claimed* one-wayness. This is because they used the same proof technique as that of [CNS02].

In this section, we show that there exists no gap between the difficulty of factoring Blum integers and the *real* one-wayness of Rabin-Paillier encryption scheme. In other words, we present the exactly tight one-wayness of Rabin-Paillier encryption scheme.

Our proof is very simple and totally elemental. In particular, no LLL algorithm is required which was essentially used in the previous proofs for RSA/Rabin-Paillier [CNS02,GMMV03].

3.1 Rabin-Paillier Encryption Scheme

Rabin-Paillier encryption scheme is described as follows. Let

$$Q_n \triangleq \{r^2 \bmod n^2 \mid r \in Z_n^*\}.$$

We say that $\bar{r} \in Z_n^*$ is *conjugate* if $(\bar{r}/n) = -1$, where (m/n) denotes Jacobi's symbol.

(Secret key) Two prime numbers p and q such that $|p| = |q|$ and $p = q = 3 \bmod 4$.

(Public key) $n(=pq), e$, where e is a prime such that $|n|/2 < e < |n|$.

(Plaintext) $m \in Z_n$.

(Ciphertext)

$$c = r^{2e} + mn \bmod n^2, \quad (1)$$

where $r \in Q_n$ is randomly chosen.

(Decryption) Since e is a prime such that $|n|/2 < e < |n|$, it satisfies that

$$\gcd(e, p-1) = \gcd(e, q-1) = 1. \quad (2)$$

Therefore, there exists d such that $ed = 1 \bmod lcm(p-1, q-1)$.

Now let $E = c^d \bmod n$. Then it is easy to see that

$$E = r^2 \bmod n.$$

We can find r such that $r \in Q_n$ uniquely because $p = q = 3 \bmod 4$. Finally, by substituting r into eq.(1), we can obtain m .

In [GMMV03], the authors showed that Rabin-Paillier encryption scheme is secure in the sense of IND-CPA if $(n, e, \mathcal{E}(n, e; 0))$ and (n, e, Q_{n^2}) are indistinguishable, where

$$\mathcal{E}(n, e; 0) \triangleq \{r^{2e} \bmod n^2 \mid r \in Q_n\}.$$

Remarks:

1. In [GMMV03], the condition on e is restricted to $\gcd(e, \lambda(n)) = 1$, where λ is Carmichael's function. However, for this parameter choice, we cannot prove that the one-wayness is as hard as the factoring problem, because we cannot generally choose such e for a given n . In Appendix B, we also point out a flaw on their claim for the semantic security of Rabin-Paillier cryptosystem.
2. RSA-Paillier encryption scheme is obtained by letting

$$c = r^e(1 + mn) \bmod n^2$$

for $m \in Z_n$ and $r \in Z_n$ [CGHN01].

3.2 Exactly Tight One-Wayness

Suppose that there exists a PPT algorithm that breaks the one-wayness with probability ε . Then Galindo et al. proved that there exists a PPT algorithm that can factor Blum integers n with probability ε^2 (see the proof of [GMMV03, Proposition 6]).

In this subsection, we show that there exists a PPT algorithm that can factor Blum integers n with probability ε . Since the converse is clear, our reduction is exactly tight.

Table 1. Factoring probability using OW-oracle with probability ε

Scheme	Factoring Probability
Galindo et al. [GMMV03]	ε^2
Our Proposed Proof	ε

Lemma 1. *Let n be a Blum integer. For any conjugate \bar{r} , there exists a unique $r \in Q_n$ such that*

$$r^2 = \bar{r}^2 \bmod n. \quad (3)$$

Further, $\gcd(r - \bar{r}, n) = p$ or q .

Proof. Note that $(-1/p) = -1$ and $(-1/q) = -1$ for a Blum integer $n = pq$. A conjugate $\bar{r} \in Z_n^*$ satisfies $(\bar{r}/n) = -1$, namely $(I) : (\bar{r}/p) = 1 \wedge (\bar{r}/q) = -1$ or $(II) : (\bar{r}/p) = -1 \wedge (\bar{r}/q) = 1$. In the case of (I) , define $r = \bar{r} \bmod p$ and $r = -\bar{r} \bmod q$, then the statement of the lemma is obtained. Similarly in the case of (II) we assign $r = -\bar{r} \bmod p$ and $r = \bar{r} \bmod q$.

Theorem 1. Rabin-Paillier encryption scheme is (t, ε) -one-way if Blum factoring problem is (t', ε) -hard, where $t' = t + \mathcal{O}((\log n)^3)$.

Proof. Suppose that there exists an oracle \mathcal{O} which breaks the one-wayness of Rabin-Paillier encryption scheme with probability ε in time t . We will show a factoring algorithm A .

We show how to find r and \bar{r} satisfying eq.(3). On input n , A first chooses a prime e such that $|n|/2 < e < |n|$ randomly. A next chooses a conjugate $\bar{r} \in Z_n^*$ and a (fake) plaintext $\bar{m} \in Z_n$ randomly, and computes a (fake) ciphertext

$$c = \bar{r}^{2e} + \bar{m}n \bmod n^2.$$

It is clear that c is uniquely written as $c = B_0 + B_1n \bmod n^2$ for some $B_0 \in Q_n, B_1 \in Z_n$. Note that

1. B_1 is uniformly distributed over Z_n because \bar{m} is randomly chosen from Z_n , and
2. B_0 is uniformly distributed over $\{r^{2e} \bmod n \mid r \in Q_n\}$ from Lemma 1.

Therefore, c is distributed in the same way as valid ciphertexts.

Now A queries c to the oracle \mathcal{O} . \mathcal{O} then answers a (valid) plaintext m such that

$$c = r^{2e} + mn \bmod n^2$$

with probability ε in time t , where $r \in Q_n$. Then we have

$$c = r^{2e} = \bar{r}^{2e} \bmod n.$$

Hence we see that $r^2 = \bar{r}^2 \bmod n$. Therefore, r^2 is written as

$$r^2 = \bar{r}^2 + yn \tag{4}$$

for some $y \in Z_n$ (with no modulus). By letting $x = \bar{r}^2 \bmod n^2$, we obtain that

$$w \triangleq c - mn = r^{2e} = (x + yn)^e = x^e + eynx^{e-1} \bmod n^2. \tag{5}$$

It is easy to see that

$$eyx^{e-1} = \frac{w - x^e}{n} \bmod n.$$

Therefore y is obtained as

$$y = (ex^{e-1})^{-1} \frac{w - x^e}{n} \bmod n.$$

Substitute y into eq.(4). Then we can compute a square root $r > 0$ because eq.(4) has no modulus. Finally we can factor n by using (r, \bar{r}) from Lemma 1. \square

Our algorithm A for Rabin-Paillier scheme is summarized as follows.

Exact_OW_Rabin_Paillier

Input: (n, e) , public key of Rabin-Paillier scheme

Output: p, q , factoring of n

1. choose a random $\bar{r} \in Z_n^*$ such that $(\bar{r}/n) = -1$.
 2. compute $x = \bar{r}^2 \bmod n^2$.
 3. choose a random (fake) plaintext $\bar{m} \in Z_n$.
 4. compute a ciphertext $c = x^e + \bar{m}n \bmod n^2$.
 5. obtain a valid plaintext $m = \mathcal{O}(c)$
 6. compute $w = c - mn = r^{2e} \bmod n^2$.
 7. compute $u = (w - x^e \bmod n^2)/n$.
 8. compute $y = u(ex^{(e-1)})^{-1} \bmod n$.
 9. compute $v = \bar{r}^2 + ny$.
 10. find $r > 0$ such that $r^2 = v$ in Z .
 11. return $\gcd(\bar{r} - r, n)$.
-

4 New Encryption Scheme

In this section, we propose an encryption scheme such that its one-wayness is as hard as the *general* factoring problem of $n = pq$ (not factoring Blum integers). The proposed scheme is obtained from an encryption scheme proposed by Kurosawa et al. [KIT88,KOMM01].

4.1 Kurosawa et al.'s Encryption Scheme

Kurosawa et al.'s showed an encryption scheme as follows [KIT88].

(Secret key) Two prime numbers p and q such that $|p| = |q|$.

(Public key) $n(= pq)$ and α such that

$$(\alpha/p) = (\alpha/q) = -1, \quad (6)$$

where (α/p) denotes Legendre's symbol.

(Plaintext) $m \in Z_n^*$.

(Ciphertext) $c = (E, s, t)$ such that

$$E = m + \frac{\alpha}{m} \bmod n \quad (7)$$

$$s = \begin{cases} 0 & \text{if } (m/n) = 1; \\ 1 & \text{if } (m/n) = -1, \end{cases} \quad t = \begin{cases} 0 & \text{if } (\alpha/m \bmod n) > m; \\ 1 & \text{if } (\alpha/m \bmod n) < m. \end{cases}$$

(Decryption) From eq.(7), it holds that

$$m^2 - Em + \alpha = 0 \bmod n. \quad (8)$$

The above equation has four roots. However, we can decrypt m uniquely from (s, t) due to eq.(6) [KIT88,KOMM01]. Also see [KT03, Appendix E].

In [KIT88,KOMM01], it is proved that this encryption scheme is one-way under the general factoring assumption.

4.2 Proposed Encryption Scheme

(Secret key) Two prime numbers p and q such that $|p| = |q|$.

(Public key) $n(= pq)$, e, α , where e is a prime such that $|n|/2 < e < |n|$ and $\alpha \in Z_n^*$ satisfies

$$(\alpha/p) = (\alpha/q) = -1. \quad (9)$$

(Plaintext) $m \in Z_n$.

(Ciphertext)

$$c = \left(r + \frac{\alpha}{r}\right)^e + mn \bmod n^2, \quad (10)$$

where $r \in Z_n^*$ is a random element such that $(r/n) = 1$ and $(\alpha/r \bmod n) > r$. (We can compute $1/r \bmod N^2$ faster than the direct method [KT03, Sec.4.3].)

(Decryption) Let $E = c^d \bmod n$, where $ed = 1 \bmod lcm(p-1, q-1)$. Then it is easy to see that

$$E = r + \frac{\alpha}{r} \bmod n.$$

Note that $(E, 0, 0)$ is the ciphertext of r by Kurosawa et al.'s encryption scheme. Therefore we can find r by decrypting $(E, 0, 0)$ with the decryption algorithm. Finally, by substituting r into eq.(10), we can obtain m .

5 One-Wayness of the Proposed Scheme

In this section, we show the one-wayness of the proposed scheme by applying our proof technique developed in Sec.3. Our security reduction is very tight. That is, there is almost no gap between the one-wayness and the hardness of the general factoring problem. Indeed, our proof requires only one decryption-oracle query while the previous proof for RSA/Rabin-Paillier encryption scheme requires two oracle queries [CNS02,GMMV03].

5.1 Proof of One-Wayness

We say that

1. $r \in Z_n^*$ is *principal* if $(r/n) = 1$ and $(\alpha/r \bmod n) > r$.
2. $\bar{r} \in Z_n^*$ is *conjugate* if $(\bar{r}/n) = -1$.

Note that in terms of the parameters of Kurosawa et al's encryption scheme, $r \in Z_n^*$ is *principal* if $(s, t) = (0, 0)$ and $\bar{r} \in Z_n^*$ is *conjugate* if $s = 1$.

Lemma 2. *For any conjugate \bar{r} , there exists a unique principal r such that*

$$E \stackrel{\Delta}{=} \bar{r} + \frac{\alpha}{\bar{r}} = r + \frac{\alpha}{r} \bmod n. \quad (11)$$

Further, $\gcd(r - \bar{r}, n) = p$ or q .

Proof. There are four different solutions of Kurosawa et al's encryption E corresponding to $(s, t) = (0, 0), (0, 1), (1, 0), (1, 1)$ as shown in [KIT88,KOMM01]. (Also see [KT03, Appendix E].) A conjugate \bar{r} satisfies $(\bar{r}/p) = 1 \wedge (\bar{r}/q) = -1$ or $(\bar{r}/p) = -1 \wedge (\bar{r}/q) = 1$ for $s = 1$. Define $r_1 = \bar{r} \bmod p \wedge r_1 = \alpha/\bar{r} \bmod q$ and $r_2 = \alpha/\bar{r} \bmod p \wedge r_2 = \bar{r} \bmod q$. Then either r_1 or r_2 is the required principle r . Hence, the former part of this Lemma holds. Further, $r \neq \bar{r} \bmod p \wedge r = \bar{r} \bmod q$ or $r = \bar{r} \bmod p \wedge r \neq \bar{r} \bmod q$ holds due to $(\alpha/p) = (\alpha/q) = -1$. Therefore, we can see that $\gcd(r - \bar{r}, n) = p$ or q . \square

From eq.(11), it holds that

$$r + \alpha/r = (\bar{r} + \alpha/\bar{r}) + yn \bmod n^2 \quad (12)$$

for some unique $y \in Z_n^*$.

Lemma 3. *Suppose that we have (\bar{r}, y) satisfying eq.(12) for some principal r , where \bar{r} is conjugate. Then we can factor n .*

Proof. We show that r can be computed from (y, \bar{r}) . Let

$$v = (\bar{r} + \alpha/\bar{r}) + yn \bmod n^2.$$

Then we have

$$r^2 - vr + \alpha = 0 \bmod n^2$$

from eq.(12). We can solve this quadratic equation by using the Coppersmith's algorithm [Cop96] because of $0 < r < n$. Then we can factor n from Lemma 2. \square

Lemma 4. Suppose that there exists an oracle \mathcal{O} that breaks the one-wayness of the proposed scheme with probability ε and in time t . Then there exists an algorithm A which factors n from (n, e, α) with probability ε in time $t + \text{poly}(\log n)$, where \mathcal{O} is invoked once.

Proof. We show how to find \bar{r} and y satisfying eq.(12). On input (n, e, α) , A first chooses a conjugate $\bar{r} \in Z_n^*$ randomly and computes

$$x = \bar{r} + \frac{\alpha}{\bar{r}} \bmod n^2. \quad (13)$$

It next chooses a (fake) plaintext $\bar{m} \in Z_n$ randomly and computes

$$c = x^e + \bar{m}n \bmod n^2.$$

It is clear that c is uniquely written as $c = B_0 + B_1n \bmod n^2$ for some $B_0, B_1 \in Z_n$. Note that (1) B_1 is uniformly distributed over Z_n because \bar{m} is randomly chosen from Z_n . (2) B_0 is uniformly distributed over $\{(r + \alpha/r)^e \bmod n \mid r \in Z_n^* \text{ is principal}\}$ from Lemma 2. Therefore, c is distributed in the same way as valid ciphertexts.

Now A queries c to the oracle \mathcal{O} . \mathcal{O} then answers a (valid) plaintext m such that

$$c = \left(r + \frac{\alpha}{r}\right)^e + mn \bmod n^2$$

with probability ε and in time t , where $r \in Z_n^*$ is principal. Then we have

$$c = \left(r + \frac{\alpha}{r}\right)^e = x^e \bmod n.$$

Hence we see that $r + \frac{\alpha}{r} = x \bmod n$. Therefore, there exists $y \in Z_n$ such that

$$r + \frac{\alpha}{r} = x + yn \bmod n^2.$$

We then obtain that

$$w \triangleq c - mn = (r + \alpha/r)^e = (x + yn)^e = x^e + eynx^{e-1} \bmod n^2.$$

It is easy to see that

$$eyx^{e-1} = \frac{w - x^e}{n} \bmod n.$$

Therefore y is obtained as

$$y = \frac{w - x^e}{n} (ex^{e-1})^{-1} \bmod n.$$

Finally we can factor n by using (\bar{r}, y) from Lemma 3. □

Our algorithm A for the proposed scheme is summarized as follows:

OW_Reciprocal_Paillier

Input: (n, e, α) , public-key of the proposed scheme

Output: p, q , factoring of n

1. choose a random $\bar{r} \in Z_n^*$ such that $(\bar{r}/n) = -1$.
 2. compute $x = \bar{r} + \alpha/\bar{r} \bmod n^2$.
 3. choose a random (fake) plaintext $\bar{m} \in Z_n^*$.
 4. compute a ciphertext $c = x^e + \bar{m}n \bmod n^2$.
 5. obtain a valid plaintext $m = \mathcal{O}(c)$
 6. compute $w = c - mn = (r + \alpha/r)^e \bmod n^2$.
 7. compute $u = (w - x^e)/n$.
 8. compute $y = u(ex^{(e-1)})^{-1} \bmod n$.
 9. compute $v = (\bar{r} + \alpha/\bar{r}) + ny \bmod n$.
 10. solve $r^2 - vr + \alpha = 0 \bmod n^2$ using Coppersmith's algorithm [Cop96].
 11. return $\gcd(\bar{r} - r, n)$.
-

Theorem 2. *The proposed encryption scheme is (t, ε) one-way if the general factoring problem is $(t', \varepsilon/2)$ -hard, where $t' = t + \text{poly}(\log n)$.*

Proof. Suppose that there exists a PPT algorithm that breaks the one-wayness of the proposed scheme with probability ε in time t . Then we show a PPT algorithm which can factor n .

For a given n , we choose a prime e such that $|n|/2 < e < |n|$ randomly. We also choose $\alpha \in Z_n^*$ such that $(\alpha/n) = 1$ randomly. It is easy to see that α satisfies eq.(9) with probability $1/2$. Next apply Lemma 4 to (n, e, α) . Then we can factor n with probability $\varepsilon/2$ in time $t' = t + \text{poly}(\log n)$. \square

The proposed scheme is a combination of the scheme of Kurosawa et al. and the RSA-Paillier scheme. Another construction is to encrypt a message $m \in Z/nZ$ as follows:

$$c = \left(r^e + \frac{\alpha}{r^e} \right) + mn \bmod n^2, \quad (14)$$

where $r \in Z_n^*$ is a random element such that $(r^e \bmod n/n) = 1$ and $(\alpha/r^e \bmod n) > r$. After computing $r^e \bmod n^2$ the reciprocal encryption is applied. However, the security analysis of this construction is more difficult — we cannot apply the above proof technique to this scheme, because $r^e \bmod n^2$ is larger than n .

5.2 Hensel Lifting and Large Message Space

Catalano et al. proved that Hensel-RSA problem is as hard as breaking RSA for any lifting index l [CNS02].

In this section, we define Hensel-Reciprocal problem and show that it is as hard as general factorization for any lifting index l . This result implies that we

can enlarge the message space of the proposed encryption scheme for $m \in Z_{n^2}$ in such a way that

$$c = r^e + mn \bmod n^l.$$

Suppose that we are given a public key (n, e, α) of the proposed encryption scheme and

$$y = \left(r + \frac{\alpha}{r}\right)^e \bmod n,$$

where $r \in Z_n^*$ is principal. The Hensel-Reciprocal problem is to compute

$$Y = \left(r + \frac{\alpha}{r}\right)^e \bmod n^l$$

from (n, e, α, y) and l , where $r \in Z_n^*$ is principal and l is a positive integer. Then we can prove the following theorem (See [KT03]).

Theorem 3. *The Hensel-Reciprocal problem is as hard as general factorization for any lifting index $l \geq 2$.*

Proof. It is easy to see that we can solve the Hensel-Reciprocal problem if we can factor n . We will prove the converse.

Suppose that there exists a PPT algorithm which can solve the Hensel-Reciprocal problem with probability ε for some $l \geq 2$. That is, the PPT algorithm can compute $Y = \left(r + \frac{\alpha}{r}\right)^e \bmod n^l$ from (n, e, α, y) and $l \geq 2$, where $r \in Z_n^*$ is principal. Then we can compute $Y' = \left(r + \frac{\alpha}{r}\right)^e \bmod n^2$. Now similarly to the proof of Lemma 4 and Theorem 2, we can factor n with probability $\varepsilon/2$ in polynomial time. \square

6 Semantic Security of the Proposed Scheme

In this section, we discuss the semantic security of the proposed scheme. Let (n, e, α) be a public key of the proposed encryption scheme.

6.1 Semantic Security

Let

$$\begin{aligned} SMALL_{RSAP}(n, e) &\triangleq \{(n, e, x) \mid x = r^e \bmod n^2, r \in Z_n\} \\ LARGE_{RSAP}(n, e) &\triangleq \{(n, e, x) \mid x = r^e \bmod n^2, r \in Z_{n^2}\} \end{aligned}$$

Note that

$$|SMALL_{RSAP}(n, e)| = n, \quad \text{and} \quad |LARGE_{RSAP}(n, e)| = n^2.$$

It is known that RSA-Paillier encryption scheme is IND-CPA if $SMALL_{RSAP}(n, e)$ and $LARGE_{RSAP}(n, e)$ are indistinguishable [CGHN01]. We call it RSA-Paillier assumption.

We now define $SMALL_{RSAK}(n, e, \alpha)$ and $LARGE_{RSAK}(n, e, \alpha)$ as follows.

$$SMALL_{RSAK}(n, e, \alpha) \triangleq \{(n, e, \alpha, x) \mid x = \left(r + \frac{\alpha}{r}\right)^e \bmod n^2, r \in Z_n^* \text{ is principal}\}$$

$$LARGE_{RSAK}(n, e, \alpha) \triangleq \{(n, e, \alpha, x) \mid x = \left(r + \frac{\alpha}{r}\right)^e \bmod n^2, r \in Z_{n^2}^*\}.$$

Note that

$$|SMALL_{RSAK}(n, e, \alpha)| = \phi(n)/4, \quad \text{and} \quad |LARGE_{RSAK}(n, e, \alpha)| = \phi(n)n/4,$$

because $r + \frac{\alpha}{r} \bmod n^2$ is a 4 : 1 mapping.

Theorem 4. *The proposed encryption scheme is secure in the sense of IND-CPA if two distributions $SMALL_{RSAK}(n, e, \alpha)$ and $LARGE_{RSAK}(n, e, \alpha)$ are indistinguishable.*

We call the above indistinguishability Reciprocal-Paillier assumption. A proof will be given in Appendix A.

6.2 Relationship with RSA-Paillier Assumption

We investigate the relationship between RSA-Paillier assumption and Reciprocal-Paillier assumption. We first generalize $SMALL_{RSAK}$ and $LARGE_{RSAK}$ so that they include α . That is, let

$$SMALL'_{RSAP}(n, e, \alpha) \triangleq \{(n, e, \alpha, x) \mid x = r^e \bmod n^2, r \in Z_n^*\}$$

$$LARGE'_{RSAP}(n, e, \alpha) \triangleq \{(n, e, \alpha, x) \mid x = r^e \bmod n^2, r \in Z_{n^2}^*\}$$

We then define *modified RSA-Paillier assumption* as follows: $SMALL'_{RSAP}(n, e, \alpha)$ and $LARGE'_{RSAP}(n, e, \alpha)$ are indistinguishable. We next define *reciprocal assumption* as follows: $SMALL_{RSAK}(n, e, \alpha)$ and $SMALL'_{RSAP}(n, e, \alpha)$ are indistinguishable.

Then we have the following corollary of Theorem 4.

Corollary 1. *The proposed encryption scheme is secure in the sense of IND-CPA if both modified RSA-Paillier assumption and the reciprocal assumption hold.*

Proof. We prove that $LARGE_{RSAK}(n, e, \alpha)$ and $LARGE'_{RSAP}(n, e, \alpha)$ are indistinguishable under the reciprocal assumption. Let \mathcal{O} be an oracle that distinguishes two distributions $LARGE_{RSAK}(n, e, \alpha)$ and $LARGE'_{RSAP}(n, e, \alpha)$. We construct a distinguisher D which can distinguish between $SMALL_{RSAK}(n, e, \alpha)$ and $SMALL'_{RSAP}(n, e, \alpha)$. For (n, e, α, c) , D chooses a random $s \in Z_n$, and computes $c' = c + ns \bmod n^2$. Then it asks (n, e, α, c') to the oracle \mathcal{O} . Because s is randomly chosen in Z_n , we can show that (n, e, α, c') is uniformly distributed

in either $LARGE_{RSAK}(n, e, \alpha)$ or $LARGE'_{RSAP}(n, e, \alpha)$. Thus the oracle \mathcal{O} can correctly distinguish between $SMALL_{RSAK}(n, e, \alpha)$ and $SMALL'_{RSAP}(n, e, \alpha)$. Therefore

$$SMALL_{RSAK} \approx SMALL'_{RSAP} \approx LARGE'_{RSAP} \approx LARGE_{RSAK},$$

where \approx means indistinguishable. This implies that Reciprocal-Paillier assumption holds. \square

7 On Chosen Ciphertext Security

For chosen ciphertext security, we can obtain a variant of our encryption scheme as follows by applying the technique of [Poi99].

$$c = ((r + \frac{\alpha}{r})^e + mn \bmod n^2) || H(r, m)$$

where H is a random hash function and $||$ denotes concatenation. In the random oracle model, (1) this scheme is one-way against chosen ciphertext attacks under the general factoring assumption. (2) It is also IND-CCA under the assumption given in Sec.6.

In the standard model, it still remains one-way and IND-CPA against chosen plaintext attacks. In general, we can prove the following theorem.

Theorem 5. *Let \mathcal{PE} be an encryption scheme with ciphertexts $c = E_{pk}(m, r)$. Suppose that (1) the set of r belongs to BPP and (2) there exists a decryption algorithm which outputs not only m but also r . For \mathcal{PE} , consider an encryption scheme $\widetilde{\mathcal{PE}}$ such that*

$$\tilde{c} = E_{pk}(m, r) || H(m, r).$$

If \mathcal{PE} is one-way against chosen plaintext attacks (IND-CPA, resp.), then $\widetilde{\mathcal{PE}}$ is one-way against chosen ciphertext attacks (IND-CCA, resp.) in the random oracle model. $\widetilde{\mathcal{PE}}$ still remains one-way against chosen plaintext attacks (IND-CPA, resp.) in the standard model.

The details will be given in the final paper.

References

- BDPR98. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among Notions of Security for Public-Key Encryption Schemes," CRYPTO'98, LNCS 1462, pp.26-45, 1998.
- Bon01. D. Boneh, "Simplified OAEP for RSA and Rabin Functions," CRYPTO 2001, LNCS 2139, pp.275-291, 2001.
- CGHN01. D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Nguyen; "Paillier's cryptosystem revisited," The 8th ACM conference on Computer and Communication Security, pp.206-214, 2001.

- CNS02. D. Catalano, P. Nguyen, and J. Stern, "The Hardness of Hensel Lifting: The Case of RSA and Discrete Logarithm," ASIACRYPT 2002, LNCS 2501, pp.299-310, 2002.
- Cop96. D. Coppersmith, "Finding a Small Root of a Univariate Modular Equation," EUROCRYPT '96, LNCS 1070, pp.155-165, 1996.
- CS98. R. Cramer and V. Shoup, "A Practical Public-Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attacks," CRYPTO'98, LNCS 1462, pp.13-25, 1998.
- CS02. R. Cramer and V. Shoup, "Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption," EUROCRYPT 2002, LNCS 2332, pp.45-64, 2002.
- FOPS01. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, "RSA-OAEP is Secure under the RSA Assumption," CRYPTO 2001, LNCS 2139, pp.260-274, 2001.
- GMMV03. D. Galindo, S. Molleví, P. Morillo, J. Villar, "A Practical Public Key Cryptosystem from Paillier and Rabin Schemes," PKC 2003, LNCS 2567, pp.279-291, 2003.
- Gol01. O. Goldreich, *Foundations of Cryptography: Basic Tools*, Cambridge University Press, 2001.
- KIT88. K. Kurosawa, T. Itoh, M. Takeuchi, "Public Key Cryptosystem using a Reciprocal Number with the Same Intractability as Factoring a Large Number," CRYPTOLOGIA, XII, pp.225-233, 1988.
- KT03. K. Kurosawa and T. Takagi, "Some RSA-based Encryption Schemes with Tight Security Reduction," A long version of this paper, IACR ePrint archive, 2003/157, 2003. (available from <http://eprint.iacr.org/>)
- KOMM01. K. Kurosawa, W. Ogata, T. Matsuo, S. Makishima, "IND-CCA Public Key Schemes Equivalent to Factoring $n=pq$, PKC 2001, LNCS 1992, pp.36-47, 2001.
- OU98. T. Okamoto and S. Uchiyama, "A New Public Key Cryptosystem as Secure as Factoring," Eurocrypt'98, LNCS 1403, pp.308-318, 1998/
- Poi99. D. Pointcheval, "New Public Key Cryptosystems based on the Dependent-RSA Problems," Eurocrypt'99, LNCS 1592, pp.239-254, 1999.
- ST02. K. Sakurai, T. Takagi, "New Semantically Secure Public-Key Cryptosystems from the RSA-Primitive," PKC 2002, LNCS 2274, pp.1-16, 2002.
- Sho01. V. Shoup, "OAEP Reconsidered," CRYPTO 2001, LNCS 2139, pp.239-259, 2001.
- Tak97. T. Takagi, "Fast RSA-Type Cryptosystems using N-adic Expansion," CRYPTO '97, LNCS 1294, pp.372-384, 1997.

A Semantic Security of the Proposed Scheme

A.1 Basic Result

Let $ZERO(n, e, \alpha)$ be the set of ciphertexts for $m = 0$ and $ALL(n, e, \alpha)$ be the set of ciphertexts for all $m \in Z_n$. That is,

$$ZERO(n, e, \alpha) \triangleq \left\{ \left(r + \frac{\alpha}{r} \right)^e \bmod n^2 \mid r \in Z_n^* \text{ is principal} \right\}$$

$$ALL(n, e, \alpha) \triangleq \left\{ \left(r + \frac{\alpha}{r} \right)^e + mn \bmod n^2 \mid m \in Z_n \text{ and } r \in Z_n^* \text{ is principal} \right\}.$$

Define

$$\begin{aligned} \text{Reciprocal}_0(n, e, \alpha) &\triangleq \{(n, e, \alpha, x) \mid x \in \text{ZERO}(n, e, \alpha)\} \\ \text{Reciprocal}_{ALL}(n, e, \alpha) &\triangleq \{(n, e, \alpha, x) \mid x \in \text{ALL}(n, e, \alpha)\} \end{aligned}$$

Note that we have $\text{Reciprocal}_0(n, e, \alpha) = \text{SMALL}_{\text{RSAK}}(n, e, \alpha)$ from their definition.

Theorem 6. *The proposed encryption scheme is secure in the sense of IND-CPA if and only if $\text{Reciprocal}_0(n, e, \alpha)$ and $\text{Reciprocal}_{ALL}(n, e, \alpha)$ are indistinguishable.*

Proof. Suppose that there exists an adversary $B = (B_1, B_2)$ which breaks our encryption scheme in the sense of IND-CPA, where B_1 works in the find stage and B_2 works in the guess stage.

We will show a distinguisher D which can distinguish between two distributions $\text{Reciprocal}_0(n, e, \alpha)$ and $\text{Reciprocal}_{ALL}(n, e, \alpha)$. Let (n, e, α, x) be the input to D , where $x \in \text{ZERO}(n, e, \alpha)$ or $x \in \text{ALL}(n, e, \alpha)$.

1. D gives $pk = (n, e, \alpha)$ to B_1 .
2. Then B_1 outputs (m_0, m_1, state) .
3. D chooses a bit b randomly and computes

$$c_b = x + m_b n \bmod n^2.$$

D gives (c_b, state) to B_2 .

4. B_2 outputs a bit \tilde{b} .
5. D outputs "0" if $\tilde{b} = b$. Otherwise, D outputs "1".

Let P_0 denote the probability that $D = 0$ for $x \in \text{ZERO}(n, e, \alpha)$ and P_{ALL} denote the probability that $D = 0$ for $x \in \text{ALL}(n, e, \alpha)$.

Now if $x \in \text{ALL}(n, e, \alpha)$, then c_b is uniformly distributed over $\text{ALL}(n, e, \alpha)$ for both $b = 0$ and 1 . Therefore, it is clear that

$$P_{ALL} = 1/2.$$

On the other hand, if $x \in \text{ZERO}(n, e, \alpha)$, then c_b is a valid ciphertext of m_b . Therefore, from our assumption and from Def.2, we obtain that

$$|P_0 - 1/2| = |\Pr(\tilde{b} = b) - 1/2|$$

is non-negligible. Hence

$$|P_0 - P_{ALL}|$$

is non-negligible because $P_{ALL} = 1/2$. This means that D can distinguish between $\text{Reciprocal}_0(n, e, \alpha)$ and $\text{Reciprocal}_{ALL}(n, e, \alpha)$.

Next suppose that there exists a distinguisher D which is able to distinguish between $\text{Reciprocal}_0(n, e, \alpha)$ and $\text{Reciprocal}_{ALL}(n, e, \alpha)$. We will show an adversary $B = (B_1, B_2)$ which breaks our encryption scheme in the sense of IND-CPA, where B_1 works in the find stage and B_2 works in the guess stage.

On input $pk = (n, e, \alpha)$, B_1 outputs $m_0 = 0$ and $m_1 \in Z_n$, where m_1 is randomly chosen from Z_n . For a given ciphertext c_b , B_2 gives (n, e, α, c_b) to D , where c_b is a ciphertext of m_b .

Note that c_0 is randomly chosen from $ZERO(n, e, \alpha)$ and c_1 is randomly chosen from $ALL(n, e, \alpha)$. Therefore, D can distinguish them from our assumption. Hence B_2 can distinguish them. \square

A.2 Extended Result

Lemma 5. $Reciprocal_{ALL}(n, e, \alpha) = LARGE_{RSAK}(n, e, \alpha)$.

Proof. First suppose that $(n, e, \alpha, c) \in LARGE_{RSAK}(n, e, \alpha)$. Then

$$c = \left(r + \frac{\alpha}{r}\right)^e \bmod n^2$$

for some $r \in Z_{n^2}^*$. Decrypt c by our decryption algorithm. Then we can find $m \in Z_n$ and a principal $r' \in Z_n^*$ such that

$$c = \left(r' + \frac{\alpha}{r'}\right)^e + mn \bmod n^2.$$

Therefore $(n, e, \alpha, c) \in Reciprocal_{ALL}(n, e, \alpha)$. This means that

$$LARGE_{RSAK}(n, e, \alpha) \subseteq Reciprocal_{ALL}(n, e, \alpha).$$

Next suppose that $(n, e, \alpha, c) \in Reciprocal_{ALL}(n, e, \alpha)$. Then

$$c = \left(r + \frac{\alpha}{r}\right)^e + mn \bmod n^2$$

for some $m \in Z_n$ and a principal $r \in Z_{n^2}^*$. We will show that there exists $u \in Z_{n^2}^*$ such that

$$c = \left(u + \frac{\alpha}{u}\right)^e \bmod n^2 \tag{15}$$

and $u \bmod n$ is principal. The above equation holds if and only if

$$u^2 - c^d u + \alpha = 0 \bmod n^2, \tag{16}$$

where $ed = 1 \bmod \phi(n)n$. For y_p such that

$$(r^2 - c^d r + \alpha) + py_p(2r - c^d) = 0 \bmod p^2,$$

let $u_p = r + py_p \bmod p^2$. Then it is easy to see that

$$u_p^2 - c^d u_p + \alpha = 0 \bmod p^2.$$

Similarly for y_q such that

$$(r^2 - c^d r + \alpha) + qy_q(2r - c^d) = 0 \bmod q^2,$$

let $u_q = r + qy_q \bmod q^2$. Then

$$u_q^2 - c^d u_q + \alpha = 0 \bmod p^2.$$

Now consider u such that

$$u = u_p \bmod p^2, \quad u = u_q \bmod q^2.$$

Then u satisfies eq.(16). Therefore u satisfies eq.(15). This means that $c \in \text{LARGE}_{\text{RSAK}}(n, e, \alpha)$. Hence

$$\text{Reciprocal}_{\text{ALL}}(n, e, \alpha) \subseteq \text{LARGE}_{\text{RSAK}}(n, e, \alpha).$$

Consequently

$$\text{LARGE}_{\text{RSAK}}(n, e, \alpha) = \text{Reciprocal}_{\text{ALL}}(n, e, \alpha).$$

□

A.3 Proof of Theorem 4

From Theorem 6 and Lemma 5, the proposed encryption scheme is IND-CPA if if $\text{Reciprocal}_0(n, e, \alpha)$ and $\text{LARGE}_{\text{RSAK}}(n, e, \alpha)$ are indistinguishable. From the definition we have $\text{Reciprocal}_0(n, e, \alpha) = \text{SMALL}_{\text{RSAK}}(n, e, \alpha)$.

B Flaw on the Semantic Security of Rabin-Paillier

Let

$$\text{SMALL}_{\text{QR}}(n, e) \triangleq \{(n, e, x) \mid x = r^{2e} \bmod n^2, r \in Q_n\}$$

$$\text{LARGE}_{\text{QR}}(n, e) \triangleq \{(n, e, x) \mid x = r^{2e} \bmod n^2, r \in Q_{n^2}\}$$

Rabin-Paillier encryption scheme is IND-CPA if and only if $\text{SMALL}_{\text{QR}}(n, e)$ and $\text{LARGE}_{\text{QR}}(n, e)$ are indistinguishable [GMMV03, Proposition 9].

Galindo et al. further claimed that $\text{SMALL}_{\text{QR}}(n, e)$ and $\text{LARGE}_{\text{QR}}(n, e)$ are indistinguishable if

- $\text{SMALL}_{\text{RSAP}}(n, e)$ and $\text{LARGE}_{\text{RSAP}}(n, e)$ are indistinguishable (RSA-Paillier is IND-CPA under this condition) and
- $\text{QR}(n)$ and $\text{QNR}(n, +)$ are indistinguishable, where

$$\text{QR}(n) \triangleq \{(n, x) \mid x \in Q_n\}$$

$$\text{QNR}(n, +) \triangleq \left\{ (n, x) \mid x \in Z_n^*, \left(\frac{x}{n} \right) = 1 \right\}$$

in [GMMV03, Proposition 11].

However, this claim is wrong. In the proof, they say that D_1 and D_2 are indistinguishable, where

$$D_1 \triangleq \{x \mid x = r^e \bmod n^2, r \in Q_n\}$$

$$D_2 \triangleq \{x \mid x = r^e \bmod n^2, r \in Z_n^*\}.$$

However, we can distinguish them easily by computing $\left(\frac{x}{n} \right)$.

A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications

Emmanuel Bresson¹, Dario Catalano², and David Pointcheval²

¹ Cryptology Department, CELAR, 35174 Bruz Cedex, France
`Emmanuel.Bresson@polytechnique.org`

² École normale supérieure, Laboratoire d'Informatique
45 rue d'Ulm, 75230 Paris Cedex 05, France
`{dario.catalano,david.pointcheval}@ens.fr`

Abstract. At Eurocrypt '02 Cramer and Shoup [7] proposed a general paradigm to construct practical public-key cryptosystems secure against adaptive chosen-ciphertext attacks as well as several concrete examples. Among the others they presented a variant of Paillier's [21] scheme achieving such a strong security requirement and for which two, independent, decryption mechanisms are allowed. In this paper we revisit such scheme and show that by considering a different subgroup, one can obtain a different scheme (whose security can be proved with respect to a different mathematical assumption) that allows for interesting applications. In particular we show how to construct a perfectly hiding commitment schemes that allows for an on-line / off-line efficiency tradeoff. The scheme is computationally binding under the assumption that factoring is hard, thus improving on the previous construction by Catalano *et al.* [5] whose binding property was based on the assumption that inverting $\text{RSA}[N, N]$ (i.e. RSA with the public exponent set to N) is hard.

1 Introduction

Secrecy of communication is clearly one of the most important goal of cryptography, therefore many secret-key and public-key cryptosystems have been proposed to solve it. It is furthermore widely admitted that the main security notion to be achieved is the semantic security [11] (a.k.a. indistinguishability of ciphertexts). Actually, a semantically secure public-key cryptosystem is not only important for secret communications, but it is also a fundamental primitive for many more complex protocols such as electronic voting, electronic auctions and secret evaluation of functions to cite some of them. However, having a "secure" cryptosystem is in general not sufficient to construct efficient solution for the above mentioned problems. In general more specific properties, such as a kind of malleability, or even homomorphic relations, are very useful to obtain practical constructions.

Roughly speaking, a public-key encryption scheme allows someone to encrypt a message for a unique recipient, the one who owns the corresponding private key (a.k.a. decryption key). But in practice, there is often a natural hierarchy, either for security or for safety reasons: the head of a group may want to be able to read any message sent to the members of the group, people may want to be able to recover the plaintexts even if they lose their private key. Therefore, it is highly desirable to provide schemes that enable to deal with intermediate scenarios, in which users are allowed to process their own data, but not those of other users.

Moreover, in practice, there are many situations on which we need more than a plain encryption function. In particular, it is often useful to have a provably secure encryption primitive that allows to perform some computation on the plaintexts without revealing them explicitly.

In this paper we propose a simple cryptosystem achieving both the above goals.

1.1 Related Work

El Gamal's scheme [8] was the first scheme based on the discrete logarithm problem, more precisely on the Diffie-Hellman problem. Furthermore, it enjoys a multiplicative homomorphic property (as the RSA cryptosystem [22]) by which one can easily obtain an encryption of $m_1 \cdot m_2$ by simply multiplying encryptions of m_1 and m_2 . This feature, however, is not very convenient for practical purposes. Indeed for many applications one may desire an efficient cryptosystem equipped with an *additive homomorphic* property, i.e. such that from encryptions of m_1 and m_2 one can obtain the encryption of $m_1 + m_2$ by simply combining the corresponding ciphertexts. The first additively homomorphic cryptosystem was proposed by Goldwasser and Micali [11] in their seminal paper on probabilistic encryption. The Goldwasser-Micali's scheme is based on quadratic residues. Given an RSA modulus N , to encrypt a bit b one chooses a pseudo-square $g \in \mathbb{Z}_N^*$ (i.e. a non quadratic residue having Jacobi symbol equal to 1) and computes $g^b r^2 \bmod N$ for random $r \in \mathbb{Z}_N^*$. The security of the cryptosystem is based on the so-called *quadratic residuosity assumption*. To improve on bandwidth Benaloh and Fisher [1,6] proposed a generalization of Goldwasser-Micali cryptosystem based on the *prime residuosity assumption*. The basic idea of their scheme is to consider \mathbb{Z}_e (instead of \mathbb{Z}_2) as underlying message space (where e is a small prime such that it divides $\phi(N)$ but e^2 does not). To encrypt a message m one then sets $g^{mr^e} \bmod N$, where, in this case, g is a non e -residue (i.e. an element whose order is a multiple of e). The main drawback of this scheme however is that decryption is rather inefficient as it requires some kind of exhaustive search to recover the message (and thus it imposes e to be very small). A more efficient variant of the Benaloh-Fischer scheme was proposed in 1998 by Naccache and Stern [18], who observed that in order to make the decryption procedure faster one can consider a value e that is not prime but instead obtained as the product of several small primes e_1, \dots, e_n such that e divides $\phi(N)$ but none of the e_i^2 's does.

At the same time a completely different approach was proposed by Okamoto and Uchiyama [20] who suggested to work on the group \mathbb{Z}_N^* where $N = p^2q$. The resulting scheme is very efficient and allows for a pretty large bandwidth (they use \mathbb{Z}_p as underlying message space), but unfortunately it is vulnerable to a simple chosen-ciphertext attack that permits to factor the modulus.

More recently Paillier [21] proposed a generalization of the Okamoto-Uchiyama cryptosystem that works in the multiplicative group $\mathbb{Z}_{N^2}^*$ and allows to consider N as a standard RSA modulus. Details of Paillier's scheme are presented below, but its basic idea is that to encrypt a message $m \in \mathbb{Z}_N$ one selects a random value y in \mathbb{Z}_N^* and sets the ciphertext as $g^m y^N \bmod N^2$ (where g is an element whose order is a multiple of N in $\mathbb{Z}_{N^2}^*$). The semantic security of the scheme is proved with respect to the *decisional N -th residuosity assumption*: given a random value $x \in \mathbb{Z}_N^*$ it is computationally infeasible to decide if there exists another element z in $\mathbb{Z}_{N^2}^*$ such that $x \equiv z^N \bmod N^2$. Paillier's scheme is more efficient (in terms of bandwidth) than all previously described schemes, moreover no adaptive chosen ciphertext attack recovering the factorization of the modulus is known. For these reasons Paillier's proposal is the best solution presented so far in terms of additively homomorphic cryptosystems.

At Eurocrypt'02 Cramer and Shoup [7] proposed a very general and beautiful methodology to obtain security against adaptive chosen-ciphertext attacks from a certain class of cryptosystems with some well-defined algebraic properties. In particular they showed how to modify Paillier's original scheme in order to achieve such a strong security goal. The resulting variant, moreover, allows for a double decryption mechanism: one can decrypt either if the factorization of the modulus is available or if some specific discrete logarithm is known.

1.2 Our Contribution

As described above all the additively homomorphic cryptosystems known so far base their security on some assumption relying on deciding residuosity.

In this paper we further investigate on the basic Cramer-Shoup variant and show that by slightly modifying the underlying structure of the scheme we obtain a new cryptosystem that allows for some more useful applications, maintaining, at the same time, all the “good” properties and with security based on a different (non residuosity-related) decisional assumption¹. Our new public-key encryption scheme, as the proposal in [7] allows for a double decryption mechanism based either on the factorization of the modulus, or on the knowledge of a discrete logarithm. The former trapdoor can be seen as the *master* one, while the latter is a *local* one: the knowledge of a discrete logarithm helps to decrypt ciphertexts which have been encrypted with a *specific* key only, while the factorization of the modulus helps to decrypt any ciphertext, whatever the key is (as long as the underlying modular group remains the same). The basic version

¹ Here, by *non-residuosity related assumption*, we mean a *decisional assumption* which claims something different from the intractability of deciding memberships in a high-residues set.

of our scheme enjoys an additive homomorphic property (similarly to the Paillier's scheme [21]). Furthermore, it is semantically secure in the standard model, based on the decisional Diffie-Hellman assumption modulo a square composite number. Thus our proposal is the first additively homomorphic cryptosystem that can be proved semantically secure with respect to a non residuosity-related decisional assumption.

We emphasize that by applying the Cramer-Shoup [7] general methodology, our scheme can be proved secure against adaptive chosen-ciphertext attacks in the standard model.

Interestingly enough, given the master key, a kind of gap group [19] appears in which the computational Diffie-Hellman problem is hard, while the corresponding decisional problem turns out to be easy — thanks to the easiness of computing the partial discrete logarithm problem (see below). This is the first gap group structure known not based on elliptic curves and pairings.

As an additional result we show how to construct a new, efficient, perfectly hiding / computationally binding commitment scheme based on factoring. A useful property of such a commitment scheme is that it allows for an on-line/off-line efficiency trade-off, by which, one may perform the most expensive part of the work, *before* knowing the message to commit to. To our knowledge no other trapdoor commitment scheme with this property, based on factoring, is known.

2 Preliminaries

2.1 Definitions and Notations

Let $N = pq$ be a safe-prime modulus, meaning with this that p and q are primes of the form $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are also primes. In the remaining of this paper, we denote by $SP(\ell)$ the sets of safe prime numbers of length ℓ . We consider $\mathbb{G} = QR_{N^2}$ the cyclic group of quadratic residues modulo N^2 . We have $\text{ord}(\mathbb{G}) = \lambda(N^2)/2 = pp'qq' = N\lambda(N)/2$, with $\lambda(N) = 2p'q'$. The maximal order of an element in this group is $N\lambda(N)/2$, and every element of order N is of the form $\alpha = (1 + kN)$.

The latter statement is not so trivial, but it will be very useful rewritten as follows: there are exactly N elements of order N in $\mathbb{Z}_{N^2}^*$, and they are all of the form $\alpha = 1 + kN$. Furthermore, since N is odd, if one denotes by t the inverse of 2 modulo N :

$$\alpha = 1 + kN = (1 + tkN)^2 \bmod N^2.$$

Therefore, they are all in \mathbb{G} too.

2.2 The Partial Discrete Logarithm Problem

Let g be an element of maximal order in \mathbb{G} . For simplicity, we assume that $g^{\lambda(N)} \bmod N^2 = (1 + N) \bmod N^2$, that is $k = 1$. Given g and $h = g^a \bmod N^2$ (for some $a \in [1, \text{ord}(\mathbb{G})]$), Paillier [21] defined the *Partial Discrete Logarithm Problem* as the computational problem of computing $a \bmod N$. We assume this

problem is difficult (without the factorization of the modulus), as stated in the following assumption.

Assumption 1 (Partial Discrete Logarithm over $\mathbb{Z}_{N^2}^*$). *For every probabilistic polynomial time algorithm \mathcal{A} , there exists a negligible function $\text{negl}()$ such that for sufficiently large ℓ*

$$\Pr \left[\mathcal{A}(N, g, h) = a \bmod N \mid \begin{array}{l} p, q \leftarrow \mathcal{SP}(\ell/2); \ N = pq; \\ g \leftarrow \mathbb{G}; \ a \leftarrow [1, \text{ord}(\mathbb{G})]; \\ h = g^a \bmod N^2; \end{array} \right] = \text{negl}(\ell).$$

Moreover Paillier proved that, when the factorization of the modulus is available, such a problem is efficiently solvable.

Theorem 2 (See [21]). *Let N be a composite modulus product of two large primes. Let \mathbb{G} be the cyclic group of quadratic residues modulo N^2 . The Partial Discrete Logarithm problem (in \mathbb{G}) cannot be harder than factoring.*

Proof. It is easy to see that we can solve the PDL problem if the factorization of N is provided, by using the following algorithm,

1. Compute $C = h^{\lambda(N)} \bmod N^2 = (1 + N)^a \bmod N^2 = (1 + aN) \bmod N^2$;
2. Return the integer $(C - 1 \bmod N^2)/N$.

□

2.3 Details of Paillier's Cryptosystem

Let $N = pq$ be an RSA modulus and g an element having order αN ($\alpha \geq 1$) in the multiplicative group $\mathbb{Z}_{N^2}^*$. To encrypt a message $m \in \mathbb{Z}_N$ Paillier proposed the following mechanism

$$\mathcal{P}_g(m, y) = g^m y^N \bmod N^2$$

for some random $y \in \mathbb{Z}_N^*$ and he proved that:

- \mathcal{P}_g is a bijection between $\mathbb{Z}_N \times \mathbb{Z}_N^*$ and $\mathbb{Z}_{N^2}^*$.
- \mathcal{P}_g is a trapdoor function equivalent to $\text{RSA}[N, N]$.
- The above encryption scheme is semantically secure against chosen-plaintext attack under the N -residuosity assumption (see [21] for details).

Since \mathcal{P}_g is a bijection, given g , for an element $w \in \mathbb{Z}_{N^2}^*$ there exists a unique pair $(c, z) \in \mathbb{Z}_N \times \mathbb{Z}_N^*$ such that $w = g^c z^N \bmod N^2$. We say that c is the *class* of w relative to g . Informally, (see [21] for more details) Paillier defined the *Computational Composite Residuosity Class Problem* as the problem of computing c given w and assumed that it is hard to solve.

2.4 The “Lite” Cramer-Shoup Variant

Let N be a product of two safe primes p and q and g an element of order $\lambda(N)$ in $\mathbb{Z}_{N^2}^*$. Such a g can be found by randomly selecting a $\mu \in \mathbb{Z}_{N^2}^*$ and setting $g = -\mu^{2N}$. It is not hard to show that this results in a generator with overwhelming probability (see [7] for more details). Then we produce the remaining part of the public key h as follows. Randomly choose a secret key $z \in [0, N^2/2]$ and set $h = g^z \bmod N^2$. (Note that for the purposes of this paper, we are considering a *very* simplified version of the Cramer-Shoup scheme, achieving semantic security *only* with respect to a passive adversary. The reader is referred to [7] for the complete solution achieving full security properties).

To encrypt a message $m \in \mathbb{Z}_N$ one chooses a random value $r \in [0, N/4]$ and computes the ciphertext (A, B) where $A = g^r \bmod N^2$ and $B = h^r(1 + mN) \bmod N^2$.

Conversely to decrypt a ciphertext (A, B) two methods are possible: either by computing $(1 + mN)$ as $B/A^z \bmod N^2$ or by using the decryption procedure described by Paillier [21] for his scheme. Note that for this second mechanism to work, knowing the value of B is sufficient. Indeed m can be retrieved from $B = h^r(1 + mN) \bmod N^2$ as follows. We denote by π the inverse of $\lambda(N) \bmod N$ (note that $\gcd(N, \lambda(N)) = 1$):

$$m = \frac{B^{\lambda(N)} - 1 \bmod N^2}{N} \cdot \pi \pmod{N} \quad \text{since} \quad B^{\lambda(N)} = 1 + m\lambda(N)N$$

2.5 The Decisional Diffie-Hellman Problem over $\mathbb{Z}_{N^2}^*$

Informally speaking, the Decisional Diffie-Hellman Problem consists, when given two random Diffie-Hellman “public keys” $A = g^a$ and $B = g^b$, in distinguishing the resulting shared key g^{ab} from a random value (see [11] for the definition of computational indistinguishability). Of course, this is to be done without possessing neither any secret keys a, b nor the factorization of the modulus.

We thus state the *Decisional Diffie-Hellman Assumption* (DDH) over a squared composite modulus of the form $N = pq$.

Assumption 3 (DDH Assumption over $\mathbb{Z}_{N^2}^*$). *For every probabilistic polynomial time algorithm \mathcal{A} , there exists a negligible function $\text{negl}()$ such that for sufficiently large ℓ*

$$\Pr \left[\mathcal{A}(N, X, Y, Z_b \bmod N) = b \mid \begin{array}{l} p, q \leftarrow \mathcal{SP}(\ell/2); \ N = pq; \\ g \leftarrow \mathbb{G}; \ x, y, z \leftarrow [1, \text{ord}(\mathbb{G})]; \\ X = g^x \bmod N^2; Y = g^y \bmod N^2; \\ Z_0 = g^z \bmod N^2; Z_1 = g^{xy} \bmod N^2; \\ b \leftarrow \{0, 1\}; \end{array} \right] - \frac{1}{2} = \text{negl}(\ell).$$

The Decisional Diffie-Hellman Assumption is related to the regular Diffie-Hellman assumption that says that given g^a and g^b one cannot compute g^{ab} in polynomial time. Clearly this assumption relies on the hardness of computing

discrete logs. Reductions in the inverse direction are not known. Interestingly enough, if the factorization of the modulus is available solving the decisional Diffie-Hellman problem (over \mathbb{Z}_{N^2}) turns out to be easy.

Theorem 4. *Let N be a composite modulus product of two large primes. Let \mathbb{G} be the cyclic group of quadratic residues modulo N^2 . The decisional Diffie-Hellman problem (in \mathbb{G}) cannot be harder than factoring.*

Proof. Assume the factorization of the modulus is provided, we are given a challenge triplet $\mathcal{G} = (g^a, g^b, g^c)$ and we have to determine if it is a Diffie-Hellman triplet or not. Our strategy is as follows. Using the factorization of the modulus we compute $a \bmod N$, $b \bmod N$ and $c \bmod N$, then we check whether the following relation holds:

$$ab \equiv c \bmod N. \quad (1)$$

Note that if \mathcal{G} is a Diffie-Hellman triplet, the relation (1) is in fact satisfied with probability 1. On the other hand if \mathcal{G} is not a Diffie-Hellman triplet, the probability that the relation (1) is verified is:

$$\Pr[ab \equiv c \bmod N \wedge ab \not\equiv c \bmod p'q'N].$$

Since a, b and c are random elements in $\mathbb{Z}_{N^2}^*$ they can be written as $a = a_1 + a_2N$, $b = b_1 + b_2N$ and $c = c_1 + c_2N$ where $a_1, a_2, b_1, b_2, c_1, c_2 \in \mathbb{Z}_N$. Thus denoting $\delta = a_2b_1 + a_1b_2 + a_2b_2N$ the above probability becomes

$$\begin{aligned} \Pr[a_1b_1 \equiv c_1 \bmod N \wedge \delta \not\equiv c_2 \bmod \phi(N)] \\ = \Pr[a_1b_1 \equiv c_1 \bmod N] \times \Pr[\delta \not\equiv c_2 \bmod \phi(N)]. \end{aligned}$$

The probability that $a_1b_1 = c_1 \bmod N$ for randomly chosen a_1, b_1 and c_1 is clearly $\frac{1}{N}$. On the other hand the probability that the event $\delta \not\equiv c_2 \bmod \phi(N)$ happens is bounded by $1 - \frac{1}{\phi(N)}$. In total the above probability can be bounded by $\frac{1}{N} - \frac{1}{\phi(N)N}$ and thus our strategy succeeds with probability approximately $1 - \frac{1}{N}$. \square

Remark 5. A *Gap-Group* is a group in which a computational problem is hard, but the corresponding decisional one is “easy”. In other words, the computational and the decisional problems are strictly separated in such a group. This implies that the corresponding *Gap-Problem* [19] is computationally hard. The first example of gap group was proposed by Joux and Nguyen in [15]. The above result shows that, when the factorization of N is provided, $\mathbb{Z}_{N^2}^*$ can be seen as a some kind of gap group for the Diffie-Hellman problem.

3 The Scheme

Our scheme can be seen as an additively homomorphic variant of the well-known El Gamal cryptosystem [8]. Let h and g be two elements of maximal order in \mathbb{G} . Note that, if h is computed as g^x , where $x \in_R [1, \lambda(N^2)]$, then x is coprime with $\text{ord}(\mathbb{G})$ with high probability, and thus h is of maximal order. The message space here is \mathbb{Z}_N .

Key Generation - Choose a random element $\alpha \in \mathbb{Z}_{N^2}^*$, a random value $a \in [1, \text{ord}(\mathbb{G})]$ and set $g = \alpha^2 \bmod N^2$ and $h = g^a \bmod N^2$. The public key is given by the triplet (N, g, h) while the corresponding secret key is a .

Encrypt - Given a message $m \in \mathbb{Z}_N$, a random pad r is chosen uniformly and at random in \mathbb{Z}_{N^2} the ciphertext (A, B) is computed as

$$A = g^r \bmod N^2 \quad B = h^r(1 + mN) \bmod N^2.$$

First Decryption Procedure - Knowing a , one can compute m as follows

$$m = \frac{B/(A^a) - 1 \bmod N^2}{N}.$$

Alternate Decryption Procedure - If the factorization of the modulus is provided, one can compute $a \bmod N$ and $r \bmod N$ as seen in the previous section. Let $ar \bmod \text{ord}(\mathbb{G}) = \gamma_1 + \gamma_2 N$, thus $\gamma_1 = ar \bmod N$ is efficiently computable. Note that

$$D = \left(\frac{B}{g^{\gamma_1}} \right)^{\lambda(N)} = \frac{(g^{ar}(1 + mN))^{\lambda(N)}}{g^{\gamma_1 \lambda(N)}} = 1 + m\lambda(N)N \bmod N^2.$$

So, still denoting by π the inverse of $\lambda(N)$ in \mathbb{Z}_N^* , one can compute m as

$$m = \frac{D - 1 \bmod N^2}{N} \cdot \pi \pmod{N}.$$

Remark 6. Note that even though the two described decryption procedures produce the same result when applied to correctly generated ciphertext they are not equivalent from a computational point of view. Indeed knowing the discrete logarithm a of h with respect to the base g in $\mathbb{Z}_{N^2}^*$ allows to decrypt any valid ciphertext generated using g and h as underlying public key. More precisely knowledge of a allows to decrypt any ciphertext generated with respect of a public key in $\{N\} \times \mathcal{G} \times \mathcal{H}$ where $\mathcal{G} \times \mathcal{H}$ is the set of the couples (g, h) such that $h = g^a \bmod N^2$. On the other hand knowing the factorization of the modulus allows to decrypt ciphertexts generated with respect to *any* public key in $\{N\} \times \mathbb{G} \times \mathbb{G}$.

Remark 7. Another interesting comparison is regarding the invalid (that is, not correctly generated) ciphertexts. Namely, if a ciphertext is not correctly generated, the fault can be detected when decrypting using the secret discrete logarithm. On the other hand, however, if the ciphertext is decrypted using the factorization of the modulus, the resulting - invalid - plaintext cannot be recognized as such. To illustrate this, consider the following example. Let (A, B) a given ciphertext, with $A \in \mathbb{G}$. Since g is a generator of \mathbb{G} there exists r , and thus K, m , such that:

$$\begin{aligned} A &= g^r \text{ where } r \in [1, \text{ord}(\mathbb{G})], \\ B &= h^r(K + mN) \text{ where } K, m \in \mathbb{Z}_N. \end{aligned}$$

If decrypted with the discrete logarithm trapdoor, this leads to a failure, since B/A^a differs from 1 mod N . Then, the incorrect encryption is detected.

Conversely if one decrypts using the factorization, one gets $a \bmod N$ and $r \bmod N$ and thus (let us denote $ar = \gamma_1 + \gamma_2 N$):

$$\begin{aligned} D &= \left(\frac{B}{g^{\gamma_1}} \right)^{\lambda(N)} = g^{ar\lambda(N) - \gamma_1\lambda(N)} (K + mN)^{\lambda(N)} = (K + mN)^{\lambda(N)} \bmod N^2 \\ &= K^\lambda + \lambda K^{\lambda-1} mN = K^\lambda + \lambda(K^{-1} \bmod N) mN \pmod{N^2} \\ &= 1 + \alpha N + mL\lambda N = 1 + (\alpha\pi + mL \bmod N)\lambda(N)N \pmod{N^2}, \end{aligned}$$

where one can write $K^{\lambda(N)} = 1 + \alpha N \bmod N^2$, $L = K^{-1} \bmod N$ and where π is the inverse of $\lambda \bmod N$. Thus, the output plaintext is $m' = \alpha\lambda^{-1} + mK^{-1} \bmod N$.

4 Security Requirements

4.1 One-Wayness

In this section we prove that the one-wayness of the scheme presented in section 3 can be related to the *Lift Diffie-Hellman* problem that we are about to define.

Let $g, X, Y, Z \in \mathbb{G}$ where $X = g^x \bmod N^2$, $Y = g^y \bmod N^2$ and $Z = g^{xy} \bmod N^2$. The well known (computational) Diffie-Hellman (modulo N^2) asks to compute Z when X, Y, g and N are provided. Similarly we define the *Lift Diffie-Hellman* problem as the one to compute Z when X, Y, g, N and $Z \bmod N$ are given. Of course it cannot be harder than the Computational Diffie-Hellman problem, but we don't know if the two problems are actually equivalent.

Definition 8 (Lift Diffie-Hellman Problem). *We say that the Lift Diffie-Hellman computational problem is hard if, for every probabilistic polynomial time algorithm \mathcal{A} , there exists a negligible function $\text{negl}()$ such that for sufficiently large ℓ*

$$\Pr \left[\begin{array}{l} \mathcal{A}(N, X, Y, Z \bmod N) \\ = Z \pmod{N^2} \end{array} \middle| \begin{array}{l} p, q \leftarrow \mathcal{SP}(\ell/2); \ N = pq; \\ g \leftarrow \mathbb{G}; \ x, y \leftarrow [1, \text{ord}(\mathbb{G})]; \\ X = g^x \bmod N^2; Y = g^y \bmod N^2; \\ Z = g^{xy} \bmod N^2; \end{array} \right] = \text{negl}(\ell).$$

Theorem 9 (One-wayness). *The scheme presented in section 3, is one-way if and only if the Lift Diffie-Hellman problem is hard.*

Proof. For $g, h \in \mathbb{G}$, let (N, g, h) be a public key, and $(A, B) = (g^r, h^r(1 + mN)) \bmod N^2$ an encryption of a random message m . If one can efficiently solve the Lift Diffie-Hellman problem then, on input $X = A = g^r$, $Y = h$ and $z = h^r(1 + mN) \bmod N = h^r \bmod N$, one can compute the quantity $Z = h^r \bmod N^2$ from which retrieving m is trivial.

Conversely if one can correctly extract m from a correctly generated ciphertext, then such a capability can be used to solve the Lift Diffie-Hellman

problem as follows. Assume we are given g , $X = g^x \bmod N^2$, $Y = g^y \bmod N^2$ and $z = g^{xy} \bmod N$. For a randomly chosen message m , we generate a ciphertext (A, B) as follows: we set the public key $(N, g, h = Y)$, $A = X$ and $B = z(1 + mN) \bmod N^2$. Our goal is to retrieve $Z = g^{xy} \bmod N^2$.

Let M be the extracted plaintext corresponding to (A, B) . We have by definition:

$$B = Z(1 + MN) = Z + ZMN = Z + (Z \bmod N)MN = Z + zMN \bmod N^2.$$

On the other hand, from the construction of B , it follows that $z + zmN = Z + zMN \bmod N^2$. Thus, we can efficiently compute $Z = z(1 + (m - M)N) \bmod N^2$. \square

With the following theorem we make explicit the relation existing between the lift Diffie-Hellman problem and the partial Discrete Logarithm problem.

Theorem 10. *If the Partial Discrete Logarithm problem is hard then so is the Lift Diffie-Hellman problem.*

Proof. The proof goes by a standard reduction argument. Assume we are given an oracle \mathcal{O} for the lift Diffie-Hellman problem that on input a triplet of the form $(X, Y, Z) = (g^x \bmod N^2, g^y \bmod N^2, g^{xy} \bmod N)$ returns the value $g^{xy} \bmod N^2$ with some non negligible probability ϵ . Our goal is to use the provided oracle to compute the partial discrete logarithm of a given challenge $h = g^{a_1 + a_2 N}$ in $\mathbb{Z}_{N^2}^*$ with respect to the base g (we assume g is a generator of the group \mathbb{G} of quadratic residues in $\mathbb{Z}_{N^2}^*$). Since g is a generator of \mathbb{G} any quadratic residue c can be written as $c = g^{r_1 + r_2 \lambda(N)}$ for some $r_1 \in \mathbb{Z}_{\lambda(N)}$ and $r_2 \in \mathbb{Z}_N$. Moreover $g^{\lambda(N)/2} = (1 + \alpha N)$ for some $\alpha \in \mathbb{Z}_N$.

Now we set $X = h$ and $Y = g^{r_1}(1 + r_2 N) \bmod N^2$ where r_1 is a random value in $[0 \dots (N + 1)/4]$, and r_2 a random element in \mathbb{Z}_N . Note that Y is not uniformly distributed over \mathbb{G} , but its distribution is statistically close to uniform (the statistical difference is of order $O(2^{-|p|})$). Finally we set $Z = X^{r_1} \bmod N$.

Observe that

$$Y = g^{r_1}(1 + r_2 N) = g^{r_1}(1 + \alpha r_2 \alpha^{-1} N) = g^{r_1 + \beta r_2 \lambda(N)/2} \pmod{N^2}$$

where $\beta = \alpha^{-1} \bmod N$.

Now we query the oracle \mathcal{O} on input (X, Y, Z) and with probability ϵ it will provide the correct answer Z' such that

$$Z' = g^{(a_1 + a_2 N)(r_1 + \beta r_2 \lambda(N)/2)} \bmod N^2 = X^{r_1} g^{a_1 \beta r_2 \lambda(N)/2} \bmod N^2$$

Thus

$$\frac{Z'}{X^{r_1}} = g^{a_1 \beta r_2 \lambda(N)/2} \bmod N^2 = (1 + a_1 r_2 N) \bmod N^2$$

from which we can get a_1 easily.

In [21] Paillier noted that when the order of g is maximal, and N is the product of two safe primes, then the partial discrete logarithm problem is equivalent to the problem of computing the composite residuosity class. This equivalence result can easily be extended to the case on which g is a generator of the group of quadratic residues modulo N^2 . This implies that, in our case, the Lift DH problem is at least as hard as the computational class problem introduced by Paillier.

4.2 Semantic Security

Theorem 11 (Semantic Security). *If Decisional Diffie-Hellman Assumption in $\mathbb{Z}_{N^2}^*$ holds, then the scheme presented in section 3, is semantically secure.*

Proof. For the sake of contradiction assume the scheme is not semantically secure. This means that there is a polynomial time distinguisher \mathcal{A} that can break semantic security. Our goal then is, given a quadruple $\mathcal{G} = (g, g^a, g^b, g^c)$, to use \mathcal{A} to decide if it is a Diffie-Hellman or a random one (i.e. if $c = ab \bmod \text{ord}(\mathbb{G})$ or not). The public key is first set as (N, g, h) where $h = g^a$; then once the adversary has chosen the messages m_0 and m_1 , we flip a bit d and we encrypt m_d as follows: $E(m_d) = (A, B)$ where $A = g^b$ and $B = g^c(1 + m_d N) \bmod N^2$.

Clearly if \mathcal{G} is a Diffie-Hellman quadruple, the above is a valid encryption of m_d and \mathcal{A} will give the correct response with non negligible advantage. On the other hand, if \mathcal{G} is not a Diffie-Hellman quadruple, we claim that even a polynomially unbounded adversary gains no information about m_d from $E(m_d)$ in a strong information-theoretic sense.

Let $c = ab + r \bmod \text{ord}(\mathbb{G})$, we can note that r is random and uniformly distributed in $[1, \text{ord}(\mathbb{G})]$ and can be written as $r_1 + r_2 \lambda(N)/2$, with $r_1, r_2 \in \mathbb{Z}_N$. The information received by the adversary (together with the public key) is of the form

$$g^b \bmod N^2, \quad g^{ab+r}(1 + m_d N) \bmod N^2$$

Let us concentrate on the second value (for the sake of simplicity let us assume that $g^{\lambda(N)/2} = (1 + N) \bmod N^2$).

$$\begin{aligned} g^{ab+r}(1 + m_d N) &= g^{ab} g^{r_1} g^{r_2 \lambda(N)/2} (1 + m_d N) \bmod N^2 \\ &= g^{ab+r_1} (1 + N)^{r_2} (1 + m_d N) \bmod N^2 \\ &= g^{ab+r_1} (1 + (r_2 + m_d)N) \bmod N^2. \end{aligned}$$

Note that, in the above relation, r_2 hides m_d perfectly and thus \mathcal{A} cannot guess d better than at random.

5 A First Application: Trapdoor Commitment

5.1 A New On-Line/Off-Line Trapdoor Commitment Scheme

In this section we present a new trapdoor commitment scheme based on the encryption function proposed in section 3. The security of the scheme can be proven to be equivalent to the hardness of factoring.

As sketched in the introduction an useful property of the proposed commitment function is that it allows for an on-line/off-line efficiency trade off, meaning with this that it becomes very efficient to compute when a preprocessing stage is allowed. On-line/off-line trapdoor commitment schemes were first proposed by [5]. In particular, to commit to a message m the sender has to compute only two modular multiplications (using a previously computed value). Such a value is completely independent of m and for this reason can be computed before even knowing to which message to commit to. Furthermore we point out that such a preprocessing step requires a single modular exponentiation. Thus even when the precomputation time is considered, our new scheme is basically as efficient as all the other trapdoor commitment schemes known in the literature.

5.2 Trapdoor Commitments

A trapdoor commitment scheme (a.k.a. *chameleon* commitment [16]) is a function with associated a pair of matching public and private keys (the latter also called the trapdoor of the commitment). The main property we want from such a function is collision-resistance: unless one knows the trapdoor, it is infeasible to find two inputs that map to the same value. On the other hand, knowledge of the trapdoor suffices to find collisions easily.

More formally, a trapdoor commitment scheme is a triplet $(\mathcal{K}, \mathcal{C}, \mathcal{D})$, where:

- \mathcal{K} is a randomized key generation algorithm. On input a security parameter k it outputs a pair of public and private keys: $\mathcal{K}(1^k) = (pk, sk)$.
- The function \mathcal{C} is the commitment function which depends on PK

$$\mathcal{C} : PK \times M \times R \longrightarrow C$$

where PK, M, R, C are the public key, message, randomness and committed values spaces respectively.

- The function \mathcal{D} is the collision-finding function,

$$\mathcal{D} : SK \times M \times R \times C \times M \longrightarrow R$$

on input the trapdoor information, a committed value (with its inputs) and a message it finds the corresponding random string. That is, given m, r and $c = \mathcal{C}(pk, m, r)$, for any message m' we have $\mathcal{D}(sk, m, r, c, m') = r'$ such that $c = \mathcal{C}(pk, m', r')$.

We require that

1. $(\mathcal{K}, \mathcal{C}, \mathcal{D})$ are functions computable in polynomial time.
2. No efficient algorithm, taking as input the public key, should be able to find, with non negligible probability, two messages $m \neq m'$ and two random values $r \neq r'$ such that $\mathcal{C}(pk, m, r) = \mathcal{C}(pk, m', r')$.
3. For any message m , the distribution $\{c = \mathcal{C}(pk, m, r)\}_{r \in R}$ has to be indistinguishable from uniform.

Note that the term “indistinguishable” above can be intended as usual in three ways: either the distributions are identical, or they are statistically indistinguishable or computationally indistinguishable (see [12]).

5.3 Previous Work on Trapdoor Commitments

The notion of trapdoor commitments was first proposed by Brassard, Chaum and Crépeau [4] in the context of zero-knowledge arguments. It is well known that trapdoor commitments can be based on the existence of claw-free trapdoor permutations [13,14].

A specific implementation based on factoring was presented in [13,14] and it requires a number of modular squarings in \mathbb{Z}_N^* which is proportional to the length of the committed message.

A famous scheme based on the hardness of computing discrete logarithms has been presented by Boyar et al. [3]. This scheme requires a full modular exponentiation (or alternatively, once again, a number of multiplications which is proportional to the length of the message).

The first commitment scheme with the on-line/off-line property was proposed by [5]. The security of such scheme is based on the hardness of inverting the RSA function (with public exponent set to N).

5.4 Our Commitment Scheme

Key Generation – The key generation algorithm, on input a security parameter ℓ produces a modulus N product of two safe primes of size $\ell/2$ together with a square h of maximal order in \mathbb{G} . The public key is given by N and h . The factorization of the modulus is the private key.

Committing a Message – To commit to a message $m \in \mathbb{Z}_N$ the sender chooses $r \in_R \mathbb{Z}_{N\lambda(N)/2}$ and sets

$$\mathcal{C}(r, m) = h^r(1 + mN) \bmod N^2.$$

Then he sends B to the receiver. Notice that the sender can compute h^r in advance and without needing to know m . Once m is provided, only two more multiplications are required to commit.

Remark 12. As already pointed out in [5] we notice that any commitment \mathcal{C} can be modified in order to obtain some on-line/off-line efficiency property. As a matter of fact such a “modified” commitment scheme \mathcal{C}' would work as follows: during the off-line stage the sender commits to a random value s with randomness r using \mathcal{C} as underlying commitment function. Let $a = \mathcal{C}(s, r)$ be the commitment value. Once m is known the sender commits to it by simply sending a and $c = m \oplus s$. The only problem with this approach is that it increases the length of the commitment. Here we denote by on-line/off-line commitment schemes those which achieve such an efficiency trade-off, *without* increasing the length of the committed value.

Theorem 13 (Security). *Under the assumption that factoring safe-prime moduli is hard the above function \mathcal{C} is a perfectly hiding trapdoor commitment scheme.*

Proof. First notice that, for any m , if r is uniformly distributed in $\mathbb{Z}_{N\lambda(N)/2}$, then $\mathcal{C}(m, r)$ is uniformly distributed in \mathbb{G} (this is because any $1 + mN$ is in \mathbb{G} , and h^r is uniformly distributed in \mathbb{G} , since h is a generator.)

Now given a commitment $\mathcal{C}(m, r) \in \mathbb{G}$ together with the corresponding (m, r) , knowing the factorization of the modulus, one can find collisions, for any message m' as follows. Let k be such that $h^{\lambda(N)} = (1 + kN) \bmod N^2$, and d the inverse of k in \mathbb{Z}_N^* . Thus we can write

$$\mathcal{C}(m, r) = h^r(1 + mN) = h^r(1 + kdmN) \bmod N^2 = h^{r+dm\lambda(N)} \bmod N^2.$$

This implies that we can find the required r' as follows

$$r' = r + (m - m')d\lambda(N) \bmod N\lambda(N)/2.$$

Finally to prove security we assume to have an algorithm \mathcal{A} that can find, on input (N, h) , two couples (m, r) and (m', r') such that $\mathcal{C}(m, r) = \mathcal{C}(m', r')$. Note that if $r = r'$ this implies that $m = m'$, thus we will assume that $r \neq r'$. From the two given couples one can write:

$$h^r(1 + mN) = h^{r'}(1 + m'N) \bmod N^2$$

and thus, letting $\Delta_r = r - r'$ and $\Delta_m = m' - m$,

$$h^{\Delta_r} = (1 + \Delta_m N) \bmod N^2.$$

Since h has order $\lambda(N)N/2$ and $(1 + \Delta_m N)$ has order (at most) N , this means that Δ_r is a multiple of $\lambda(N)/2$. This is enough to factor [17]. \square

5.5 Application to On-Line/Off-Line Signatures

On-line/Off-line signatures were introduced by Even, Goldreich and Micali [9]. The basic idea is to split the signature generation process in two stages: the first one, more expensive, is computed off-line before the message to sign is known. The second, more efficient, phase is performed once the message is available. The proposed method, however, is not very practical as it increases the length of the signature by a quadratic factor. More recently Shamir and Taumann [23] introduced a new paradigm — as well as several efficient constructions — based on chameleon commitments, which performs the above conversion more efficiently. Moreover, this technique, improves on the security of the underlying signature scheme which is used to sign only random strings chosen off-line by the signer.

The basic idea is as follows. During the off-line phase the signer computes a chameleon commitment function on input a random message m' and random string r' and signs the resulting value $H(m', r')$. Once the message m to sign is known, the signer use his knowledge of the trapdoor key to compute a value r such that $H(m, r) = H(m', r')$.

Using our new commitment scheme one can obtain a simple on-line/off-line signature scheme based on factoring.

6 Variants and Other Applications

6.1 A Variant of the Cryptosystem

We propose a variant of our scheme in which the randomness is chosen in a smaller set, namely in \mathbb{Z}_N rather than in \mathbb{Z}_{N^2} . Note, however, that we still consider an element g of maximal order in \mathbb{G} . To encrypt a message $m \in \mathbb{Z}_N$, the operations to perform remain the same:

$$A = g^r \bmod N^2, \quad B = h^r(1 + mN) \bmod N^2$$

With this variant, the decryption procedure that makes use of the factorization is simplified, and in particular allows to detect some incorrectly generated ciphertext. More precisely, it becomes possible to check whether the underlying random exponent r belongs to the correct interval: before decrypting a ciphertext, the receiver first recovers $\rho = \log_g A \bmod N$ using the factorization of the modulus; after that, it checks if $A = g^\rho \bmod N^2$ holds. If the equality does not hold, it rejects.

Of course, if the ciphertext is correctly generated, that is, $r \in \mathbb{Z}_N$, the recovered value ρ is actually r itself, and thus the equality holds. Whereas if A is not correctly generated, the relation $A = g^\rho$ holds with negligible probability only.

Note that decrypting such a ciphertext using the first decryption procedure (i.e., with the discrete logarithm of h to the base g), the decryption never “fail” at this step, simply because the receiver does not recover the value of r , and cannot check its range.

The decryption procedure continues as follows. If using the discrete logarithm trapdoor, the receiver computes h^r as $A^a \bmod N^2$; if using the factorization of N , he computes h^r as $h^\rho \bmod N^2$. Then in both cases, one checks whether $B/h^r = 1$ or not, and if yes, one recovers the plaintext.

6.2 The Small Diffie-Hellman Problem over $\mathbb{Z}_{N^2}^\star$

We introduce a new variant of the Diffie-Hellman Problem. In a nutshell, when given $(A, B) = (g^a, g^b)$ where b is small, i.e. $b \in \mathbb{Z}_N$, the computational (resp., decisional) problem consists in computing (resp., distinguishing from a random element in \mathbb{G}) the value $C = g^{ab} \bmod N^2$.

We thus state the *Small Decisional Diffie-Hellman Assumption* (S-DDH) over a squared composite modulus of the form $N = pq$.

Assumption 14 (Small-DDH Assumption over $\mathbb{Z}_{N^2}^\star$). *For every probabilistic polynomial time algorithm \mathcal{A} , there exists a negligible function $\text{negl}()$ such that for sufficiently large ℓ*

$$\Pr \left[\mathcal{A}(N, X, Y, Z_b \bmod N) = b \mid \begin{array}{l} p, q \leftarrow \mathcal{SP}(\ell/2); \ N = pq; \\ g \leftarrow \mathbb{G}; x, z \leftarrow [1, \text{ord}(\mathbb{G})]; y \leftarrow \mathbb{Z}_N; \\ X = g^x \bmod N^2; Y = g^y \bmod N^2; \\ Z_0 = g^z \bmod N^2; Z_1 = g^{xy} \bmod N^2; \\ b \leftarrow \{0, 1\}; \end{array} \right] - \frac{1}{2} = \text{negl}(\ell)$$

One easily proves the following two theorems:

Theorem 15. *The Small (Computational) Diffie-Hellman Problem cannot be harder than factoring.*

Theorem 16. *The above variant of our cryptosystem is semantically secure under the Small Decisional Diffie-Hellman assumption.*

Indeed, knowing the factorization of N allows to fully retrieve the second exponent, thus making the computational problem trivial. The proof for second theorem is similar to the proof for the basic scheme (theorem 11).

6.3 A New Hierarchical Encryption Scheme

A hierarchical encryption scheme [10] can be simply based on our scheme by providing the authority with the master key (the factorization of the modulus) and by giving to each player a local key (an El Gamal-like private key.)

In such a scheme, anybody is able to encrypt a message for a particular player, in such way that only this player and the authority are able to decrypt properly. Moreover, by randomly choosing two elements g, h and encrypting with respect to such a “key”, it is possible to design ciphertexts that can be decrypted by nobody but the authority.

Further work might consists in investigate such possibilities in the contexts of electronic voting or digital auctions.

7 Conclusion

This paper is a further investigation within the family of homomorphic cryptosystems modulo a squared composite number. As a first contribution, we provided a new variant of the Cramer-Shoup scheme whose main feature is to offer two different decryption procedures, based on two different trapdoors. In particular, this scheme is the first additively homomorphic cryptosystem whose security is not based on a residuosity-related assumption. Derived from this scheme is a new trapdoor commitment, whose security provably relies on the factorization problem. This commitment scheme allows for a very interesting on-line/off-line efficiency trade-off, without increasing the length of the commitment.

References

1. J. Benaloh. Verifiable Secret-Ballot Elections. PhD Thesis, Yale University, 1987.
2. D. Boneh. The decision Diffie-Hellman problem. In *Proc. of the 3rd ANTS*, LNCS 1423, pp. 48–63, Springer-Verlag, June 1998.
3. J.F. Boyar, S.A. Kurtz, and M.W. Krentel. A Discrete Logarithm Implementation of Perfect Zero-Knowledge Blobs. *Journal of Cryptology*, 2(2):63–76, 1990.
4. G. Brassard, D. Chaum, and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences*, 37, 1988.

5. D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Q. Nguyen. Paillier's Cryptosystem Revisited. In *Proc. of the 8th CCS*, pages 206–214. ACM Press, New York, 2001.
6. J. Cohen, M. Fisher. A robust and Verifiable cryptographically secure election scheme. In *Proc. of the 26th FOCS*. IEEE, 1985.
7. R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Eurocrypt '02*, LNCS 2332, pages 45–64. Springer-Verlag, Berlin, 2002.
8. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.
9. S. Even, O. Goldreich and S. Micali. On-line/Off-line Digital Signatures. In *Crypto '89*, pages 263–277. Springer-Verlag, Berlin, 1989.
10. C. Gentry and A. Silverberg. Hierarchical ID-Based Encryption. In *Asiacrypt '02*, LNCS 2501, pages 548–566. Springer-Verlag, Berlin, 2002.
11. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
12. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proc. of the 17th STOC*, pages 291–304. ACM Press, New York, 1985.
13. S. Goldwasser, S. Micali, and R. Rivest. A “Paradoxical” Solution to the Signature Problem. In *Proc. of the 25th FOCS*, pages 441–448. IEEE, New York, 1984.
14. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
15. A. Joux and K. N. Guyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology eprint archive. <http://eprint.iacr.org/2001/003/>, 2001.
16. H. Krawczyk and T. Rabin. Chameleon Hashing and Signatures. In *Proc. of NDSS '2000*. Internet Society, 2000.
17. G. Miller. Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Sciences*, 13:300–317, 1976.
18. D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *Proc. of 5th Symposium on Computer and Communications Security*. ACM, 1998.
19. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT – RSA '01*, LNCS 2020, pages 159–175. Springer-Verlag, Berlin, 2001.
20. T. Okamoto and S. Uchiyama. The Gap-Problems: A new class of problems for the security of cryptographic schemes. In *Proc. of PKC '01*, volume 1992 of LNCS. IACR, Springer-Verlag, 1998.
21. P. Paillier. Public-Key Cryptosystems Based on Discrete Logarithms Residues. In *Eurocrypt '99*, LNCS 1592, pages 223–238. Springer-Verlag, Berlin, 1999.
22. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
23. A. Shamir and Y. Taumann. Improved On-line/Off-line Signature Schemes. In *Crypto '01*, LNCS 2139, pages 355–367. Springer-Verlag, Berlin, 2001.

A Details for Theorem 10

In that theorem we use the fact that the distribution of the oracle input is statistically close from the uniform one. Here we prove this fact with more details.

More formally, we want to evaluate the statistical distance δ between the two following distributions:

$$\left\{ g^{r_1+r_2\lambda/2} \middle| (r_1, r_2) \in \mathbb{Z}_{\lambda/2} \times \mathbb{Z}_N \right\} \text{ and } \left\{ g^{r_1(1+r_2N)} \middle| (r_1, r_2) \in \mathbb{Z}_{\frac{N+1}{4}} \times \mathbb{Z}_N \right\}$$

First we note that the map $\mathbb{Z}_{\lambda/2} \times \mathbb{Z}_N \rightarrow \mathbb{G} : (c_1, c_2) \mapsto c = g^{c_1+c_2\lambda/2} \bmod N^2$ is a bijection. Thus we have to compute:

$$\begin{aligned} \delta &= \sum_{c \in \mathbb{G}} \left| \Pr_{\substack{r_1 \in \mathbb{Z}_{\lambda/2} \\ r_2 \in \mathbb{Z}_N}} \left[g^{r_1+r_2\lambda/2} = c \right] - \Pr_{\substack{r_1 \in \mathbb{Z}_{(N+1)/4} \\ r_2 \in \mathbb{Z}_N}} \left[g^{r_1(1+r_2N)} = c \right] \right| \\ &= \sum_{c \in \mathbb{G}} \left| \Pr_{r_1 \in \mathbb{Z}_{\lambda/2}} [r_1 = c_1] \Pr_{r_2 \in \mathbb{Z}_N} [r_2 = c_2] - \Pr_{\substack{r_1 \in \mathbb{Z}_{(N+1)/4} \\ r_2 \in \mathbb{Z}_N}} \left[g^{r_1(1+r_2N)} = c \right] \right| \\ &= \sum_{c \in \mathbb{G}} \left| \frac{2}{\lambda} \times \frac{1}{N} - \Pr_{\substack{r_1 \in \mathbb{Z}_{(N+1)/4} \\ r_2 \in \mathbb{Z}_N}} \left[g^{r_1(1+r_2N)} = c \right] \right| \end{aligned}$$

Denoting $g^{\lambda/2} = 1 + \alpha N \bmod N^2$ and $\beta = \alpha^{-1} \bmod N$, we have $g^{r_1(1+r_2N)} = g^{r_1+r_2\beta\lambda/2} \bmod N^2$. Then we observe that for $\lambda/2 \leq r_1 < \frac{N+1}{4}$, we have the following “collision”:

$$g^{r_1+r_2\beta\lambda/2} = g^{(r_1-\lambda/2)+(r_2\beta+1)\lambda/2} \pmod{N^2}$$

Hence, two cases appear when summing up (of course, the probabilities that r_2 or $r_2\beta$ or $r_2\beta + 1$ equals a given c_2 are all $1/N$):

$$\Pr \left[g^{r_1+r_2\beta\lambda/2} = g^{c_1+c_2\lambda/2} \right] = \begin{cases} 2 \cdot \frac{4}{N+1} \times \frac{1}{N} & \text{if } 0 \leq c < \frac{N+1}{4} - \frac{\lambda}{2} \\ 1 \cdot \frac{4}{N+1} \times \frac{1}{N} & \text{if } \frac{N+1}{4} - \frac{\lambda}{2} \leq c < \frac{\lambda}{2} \end{cases}$$

Consequently, we gets (recall that $\frac{N+1}{4} - \frac{\lambda}{2} = \frac{p+q}{4}$):

$$\delta = \frac{p+q}{4} \left| \underbrace{\frac{2}{\lambda N} - \frac{8}{N(N+1)}}_{\leq 0} \right| + \left(\frac{\lambda}{2} - \frac{p+q}{4} \right) \left| \underbrace{\frac{2}{\lambda N} - \frac{4}{N(N+1)}}_{\geq 0} \right|$$

This is easily seen negligible. □

Factoring Estimates for a 1024-Bit RSA Modulus

Arjen Lenstra¹, Eran Tromer², Adi Shamir², Wil Kortsmit³,
Bruce Dodson⁴, James Hughes⁵, and Paul Leyland⁶

¹ Citibank, N.A. and Technische Universiteit Eindhoven
1 North Gate Road, Mendham, NJ 07945-3104, USA
`arjen.lenstra@citigroup.com`

² Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot 76100, Israel
`{tromer,shamir}@wisdom.weizmann.ac.il`

³ Technische Universiteit Eindhoven
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
`rcwil@win.tue.nl`

⁴ Lehigh University, Bethlehem, PA 18015-3174, USA
`bad0@lehigh.edu`

⁵ Storage Technology Corporation
7600 Boone Ave No, Minneapolis, MN 55428, USA
`James.Hughes@StorageTek.com`

⁶ Microsoft Research Ltd
7 JJ Thomson Avenue, Cambridge, CB3 0FB, UK
`pleyland@microsoft.com`

Abstract. We estimate the yield of the number field sieve factoring algorithm when applied to the 1024-bit composite integer RSA-1024 and the parameters as proposed in the draft version [17] of the TWIRL hardware factoring device [18]. We present the details behind the resulting improved parameter choices from [18].

Keywords: 1024-bit RSA, factorization, number field sieve, TWIRL

1 Introduction

RSA with 1024-bit moduli is widely used. It is unlikely that breaking a single 1024-bit RSA modulus will change much, just as repeatedly breaking DES had, for obvious economic reasons, limited effect on legacy applications. Nevertheless, despite the possible lack of immediate practical relevance, in cryptographic circles there is wide-spread interest in the question how hard it would be to factor a 1024-bit RSA modulus (cf. [2], [12]).

At the Asiacrypt 2002 rump session an innovative hardware device, ‘TWIRL’, was presented that would be able to factor 1024-bit RSA moduli at a much lower cost than before. The work reported here was inspired by that presentation and the draft of TWIRL [17]. The draft presents cost estimates for a number field sieve (NFS) factorization of a 1024-bit composite that rely on extrapolations of parameter settings used for a 512-bit NFS factorization (cf. Section 4). To our

knowledge the accuracy of long range extrapolation from 512 to 1024 bit parameter selection had never been properly tested. Our goal was therefore to do a ‘reality check’ of the choices made in [17]. Given the many uncertainties involved in the factoring process we did not expect conclusive results but hoped to get an indication if the proposed parameters looked ‘reasonable’ or not. As it turned out, our results suggested that the choices were over-optimistic. Our approach was subsequently adopted by the authors of TWIRL. It allowed them to derive realistic parameters and to fine-tune the improved design [18]. The additional cost of the new choices is offset, approximately, by the greater efficiency of the new design, so that the overall cost estimates of [17] and [18] are similar. The details of the parameter settings from [18] are presented in Appendix B.

A sketch of our approach follows. We assume elementary background on the NFS (cf. Section 2). We selected the number RSA-1024 from [16] as a representative 1024-bit RSA modulus. This choice was supported by experiments that did not reveal significant differences between RSA-1024 and several other 1024-bit products of randomly selected 512-bit primes. We followed the search strategy from [13], [14], [15] to select number fields of degrees 5, 6, 7, 8, and 9 for RSA-1024, but we did not spend as much time on the search as we would have done for an actual factoring attempt. The resulting number fields can thus be regarded as somewhat worse than the number fields that would result from a more extensive search and the resulting estimates are on the pessimistic side. The better polynomial selection program of Jens Franke and Thorsten Kleinjung can handle only degree 5. It was used in Appendix B.

For all these number fields and a wide range of factor base sizes and sieving regions (including the choices made in [17]) we estimated the expected number of relations using numerical approximation of the applicable smoothness and semi-smoothness probabilities. Unfortunately, there is no a priori way to evaluate how close the resulting estimates are to the actual yield. To validate the estimates, we therefore ran extensive (semi-)smoothness tests on the actual numbers that would appear in an NFS factoring attempt, restricted to the most promising degrees and subsets of the sieving regions. We used the relatively slow test described in Section 3. This posed no problems because our object was determining the yield, not optimizing the speed. It can be seen in Section 5 that although the different methods do not produce identical results, the actual smoothness tests do inspire a high level of confidence in the numerical approximations.

Furthermore, we computed similar estimates for the multiple number field approach from [5], under the untested and possibly over-optimistic assumption that all number fields are about equally ‘good’ as the number fields we generated (cf. Section 6). In the same section we estimated the yield under the assumption that we are able to find much better number fields than we found, for instance by adapting the Franke/Kleinjung program to higher degrees. Corresponding actual smoothness experiments were not performed for these variations, because they involve number fields that we did not actually manage to construct.

There is nothing new to our approach and neither are the results earth-shaking. In particular we did not attempt to address the uncertainties referred

to above, namely to analyse the cycle-matching behavior of relations involving large primes. We are not aware of any progress in that area. Despite the lack of innovative results, we hope that the approach presented in this paper is helpful to other researchers in this field. From that point of view our work already proved useful, as witnessed by the evolution of [17] into [18] (cf. Appendix B).

2 Number Field Sieve Background

This section describes the parts of the number field sieve factoring algorithm which are relevant for this paper. See [10] for further details. The number of primes $\leq x$ is denoted by $\pi(x)$. An integer is *y-smooth* if all its prime factors are $\leq y$. An integer k is (y, z, ℓ) -*semi-smooth* if it is *y-smooth* except for at most ℓ prime factors that are $> y$ and $\leq z$ (referred to as *large primes*). If this is the maximal such ℓ , then k is *strictly* (y, z, ℓ) -*semi-smooth*.

Regular NFS. Let n be the number to be factored. Fix a degree d . Find an integer m (close to $n^{1/(d+1)}$), an irreducible polynomial $f \in \mathbf{Z}[X]$ of degree d such that $f(m) \equiv 0 \pmod n$, and a corresponding skewness ratio s (cf. [13], [14], [15]). This f is chosen such that the values $b^d f(a/b)$, for coprime pairs of integers (a, b) with $b > 0$, have a larger than average *y-smooth* factor, for small y . For integer k , let $\eta(y, k)$ denote the largest *y-smooth* factor of k and $\lambda(y, k) = \log(\eta(y, k))$ the natural logarithm thereof. For random integers, the expected value $E(y)$ of $\lambda(y, k)$ is known to be

$$E(y) = \sum_{p < y, p \text{ prime}} (\log p)/(p - 1).$$

The expected value $E_f(y)$ of $\lambda(y, b^d f(a/b))$ can be determined experimentally by averaging $\lambda(y, b^d f(a/b))$ over a large random set of coprime pairs (a, b) with $b > 0$. The correction factor that measures f 's advantage is defined as $t = \exp(E_f(2^{30}) - E(2^{30}))$.

Fix rational smoothness and semi-smoothness bounds y_r and z_r and algebraic ones y_a and z_a , with $y_r \leq z_r$ and $y_a \leq z_a$. Fix the number of large primes on the rational side ℓ_a and on the algebraic side ℓ_r . In the sieving step find *relations*: pairs of coprime integers (a, b) with $b > 0$ such that the *rational norm* $N_r(a, b) = |a - bm|$ is (y_r, z_r, ℓ_r) -semi-smooth and the *algebraic norm* $N_a(a, b) = |b^d f(a/b)|$ is (y_a, z_a, ℓ_a) -semi-smooth. If $N_r(a, b)$ is y_r -smooth and $N_a(a, b)$ is y_a -smooth, the relation is referred to as a full relation, otherwise it is called a partial relation. Approximately $\pi(\min(y_r, y_a))/d!$ full relations are free, namely one for each prime $p \leq \min(y_r, y_a)$ such that f has d roots modulo p (cf. [10]). A non-free relation (a, b) for which $N_r(a, b)$ is strictly (y_r, z_r, L_r) -semi-smooth and $N_a(a, b)$ is strictly (y_a, z_a, L_a) -semi-smooth will be called an (L_r, L_a) -*partial relation*. We use the standard abbreviations **ff** for $(0, 0)$ -partial relations, **fp** for $(0, 1)$ -partial relations, **pf** for $(1, 0)$ -partial relations and **pp** for $(1, 1)$ -partial relations.

For the $N_r(a, b)$'s the sieving step involves sieving with the primes $\leq y_r$, the *rational factor base* of cardinality $\pi(y_r)$. For the $N_a(a, b)$'s it involves sieving

with pairs (p, r) with $p \leq y_{\mathbf{a}}$ prime and $f(r) \equiv 0 \pmod p$, the *algebraic factor base* of cardinality $\approx \pi(y_{\mathbf{a}})$. Let $T(y_{\mathbf{r}}, y_{\mathbf{a}}) = \pi(y_{\mathbf{r}}) + \pi(y_{\mathbf{a}}) - \pi(\min(y_{\mathbf{r}}, y_{\mathbf{a}}))/d!$.

The purpose of the sieving step is to find approximately $T(y_{\mathbf{r}}, y_{\mathbf{a}})$ independent cycles: sets C of relations such that $\prod_{(a,b) \in C} N_{\mathbf{r}}(a, b)$ is a square times a $y_{\mathbf{r}}$ -smooth number and, simultaneously, $\prod_{(a,b) \in C} N_{\mathbf{a}}(a, b)$ is a square times a $y_{\mathbf{a}}$ -smooth number. The condition on the last square is slightly more involved; see below. A full relation is a cycle of length 1. Two $(1, 0)$ -partial relations whose rational norms share a large prime can be combined into a cycle of length 2. Similarly, for two $(0, 1)$ -partial relations (a_1, b_1) and (a_2, b_2) whose algebraic norms share the large prime p , a length 2 cycle follows if the relations correspond to the same root of $f \pmod p$, i.e., if $a_1/b_1 \equiv a_2/b_2 \pmod p$. Longer cycles may be built by pairing matching rational large primes or matching algebraic large primes with corresponding roots.

The part of the (a, b) -plane where relations are sought, the sieving region, consists of a, b with $-A < a \leq A$ and $0 < b \leq B$ for sufficiently large $A, B > 0$ with $A/B \approx s$. The size $2AB$ of the sieving region is denoted by S . A rectangular sieving region is in general not optimal in the sense that certain carefully chosen and somewhat smaller regions may yield the same number of relations (cf. [20]). For our yield computations this is hardly a concern.

Given approximately $T(y_{\mathbf{r}}, y_{\mathbf{a}})$ independent cycles, the factorization of n follows by applying the matrix step to the cycles and the square-root step to the results of the matrix step; these final two steps are not discussed in this paper.

Cycle Yield. The number of relations required to obtain $T(y_{\mathbf{r}}, y_{\mathbf{a}})$ independent cycles is determined by the matching behavior of the large primes. This behavior varies from factorization to factorization and is not yet well understood. Obviously, $T(y_{\mathbf{r}}, y_{\mathbf{a}})$ distinct (non-free) full relations suffice, but this is necessary only if the large primes cannot be paired at all — that has never occurred in practice so far. Furthermore, the behavior gets considerably more complicated if more than a single large prime is allowed in the rational and algebraic norms. This is customary in current factorizations because it leads to a considerable speedup (cf. [4]). The uncertainty about the matching behavior of the large primes is the main reason that it is currently impossible to give reliable estimates for the difficulty of factoring numbers that are much larger than the numbers we have experience with. For that reason, we mostly restrict ourselves to estimates of the sieving region that would be required to find $T(y_{\mathbf{r}}, y_{\mathbf{a}})/c$ non-free full relations for a range of $y_{\mathbf{r}}$ and $y_{\mathbf{a}}$ values and several values of $c \geq 1$. Note that, for any number of large primes per relation, $\pi(z_{\mathbf{r}}) + \pi(z_{\mathbf{a}})$ relations always suffice.

Effort Required. For smoothness bounds $y_{\mathbf{r}}$ and $y_{\mathbf{a}}$, sieving region size S and assuming a traditional implementation, the sieving effort is dominated by the number of times the primes and (prime, root) pairs in the factor bases hit the sieving region. This value is approximately proportional to

$$S(\log \log(y_{\mathbf{r}}) + \log \log(y_{\mathbf{a}})).$$

Furthermore, memory for the sieve and the factor bases may be needed.

Coppersmith's Multi-polynomial Version. As shown in [5] an improvement of the regular NFS can be obtained by considering a set G of irreducible degree d polynomials with shared root m modulo n . In that case, a relation is a pair of coprime integers (a, b) with $b > 0$ such that $N_{\mathbf{r}}(a, b)$ is $(y_{\mathbf{r}}, z_{\mathbf{r}}, \ell_{\mathbf{r}})$ -semi-smooth and $b^d g(a/b)$ is $(y_{\mathbf{a}}, z_{\mathbf{a}}, \ell_{\mathbf{a}})$ -semi-smooth for a $g \in G$. The goal is to find $\pi(y_{\mathbf{r}}) + \#G(\pi(y_{\mathbf{a}}) - \pi(\min(y_{\mathbf{r}}, y_{\mathbf{a}}))/d!)$ cycles. First, sieving is used to find a set V of $(y_{\mathbf{r}}, z_{\mathbf{r}}, \ell_{\mathbf{r}})$ -semi-smooth rational norms (with a and b coprime). Next, a smoothness test different from sieving is used (in [5] the elliptic curve method is suggested) to test $b^d g(a/b)$ for $(y_{\mathbf{a}}, z_{\mathbf{a}}, \ell_{\mathbf{a}})$ -semi-smoothness for all $(a, b) \in V$ and all $g \in G$. The approximate runtime of the relation collection becomes proportional to

$$S \log \log(y_{\mathbf{r}}) + E(\#V)(\#G)$$

where E is a constant of proportionality that depends on the $(y_{\mathbf{a}}, z_{\mathbf{a}}, \ell_{\mathbf{a}})$ -semi-smoothness test used. Its value is best determined empirically.

3 Number Field Sieve Analysis and Estimates

Let the notation be as above. This section describes the methods we used to estimate the yield of the NFS. Let $L_x[r, \alpha]$ denote any function of x that equals

$$\exp((\alpha + o(1))(\log x)^r (\log \log x)^{1-r}), \text{ for } x \rightarrow \infty,$$

where α and r are real numbers with $0 \leq r \leq 1$ and logarithms are natural.

Estimating Smoothness and Semi-smoothness Probabilities. Let $\sigma_{\ell}(u, v)$ denote the probability that a random integer $\leq x$ is strictly $(x^{1/u}, x^{1/v}, \ell)$ -semi-smooth, for $x \rightarrow \infty$. In particular, $\sigma_0(u, v)$ is the probability of $x^{1/u}$ -smoothness, and equals the Dickman $\rho(u)$ function (cf. [1], [6]) which is $u^{-u+o(1)}$ for $u \rightarrow \infty$ (cf. [3], [7]). Also, let $\bar{\sigma}_2(u, v, w)$ be the probability that a random integer $\leq x$ is $x^{1/u}$ -smooth except for exactly two prime factors $> x^{1/u}$ and $\leq x^{1/v}$ whose product is $< x^{1/w}$ (note that $\sigma_2(u, v) = \bar{\sigma}_2(u, v, v/2)$). We assume that these functions give good approximations of the semi-smoothness probabilities for the finite values of x that we consider (cf. Section 5, [1], [9]).

Closed expressions for σ_{ℓ} are not known. Thus, for ρ and σ_1 we used the numerical approximation methods given in [1]. To compute σ_2 and $\bar{\sigma}_2$ we used a natural generalization of [9, Theorem 3.1] and performed the integration numerically using the *GNU Scientific Library*.

Asymptotic Runtime. It is heuristically assumed that with respect to smoothness properties $N_{\mathbf{r}}(a, b)$ and $N_{\mathbf{a}}(a, b)$ behave independently as random integers of comparable sizes. It follows that a pair of coprime integers (a, b) leads to a full relation with probability $u_{\mathbf{r}}^{-u_{\mathbf{r}}+o(1)} u_{\mathbf{a}}^{-u_{\mathbf{a}}+o(1)}$, where $u_{\mathbf{r}} = \frac{\log(N_{\mathbf{r}}(a, b))}{\log(y_{\mathbf{r}})}$ and $u_{\mathbf{a}} = \frac{\log(N_{\mathbf{a}}(a, b))}{\log(y_{\mathbf{a}})}$. Optimization of the parameters leads to the heuristic asymptotic expected NFS runtime $L_n[1/3, (64/9)^{1/3}] \approx L_n[1/3, 1.923]$, for $n \rightarrow \infty$, $y_{\mathbf{r}}$ and $y_{\mathbf{a}}$ both equal to $L_n[1/3, (8/9)^{1/3}]$ (the ‘square-root of the runtime’), and the sieving region size $S = L_n[1/3, (64/9)^{1/3}]$. The correction factor t and large

primes are believed to affect these values only by a constant factor (which disappears in the $o(1)$). Coppersmith's multi-polynomial variant [5] runs, asymptotically, slightly faster in expected time $L_n[1/3, 1.902]$. These expressions provide some insight into parameter selection, but the presence of the $o(1)$ limits their practical value. See Section 4 for how they are often used in practice.

Estimating the Yield Using ρ and σ_ℓ . For actual yield estimates we include the correction factor t defined in Section 2. Redefine $u_{\mathbf{a}} = \frac{\log(N_{\mathbf{a}}(a,b)/t)}{\log(y_{\mathbf{a}})}$, and define $v_{\mathbf{r}} = \frac{\log(N_{\mathbf{r}}(a,b))}{\log(z_{\mathbf{r}})}$, $v_{\mathbf{a}} = \frac{\log(N_{\mathbf{a}}(a,b)/t)}{\log(z_{\mathbf{a}})}$. Then under the same assumptions as above, it follows that (a, b) forms an $(L_{\mathbf{r}}, L_{\mathbf{a}})$ -partial relation with probability

$$\sigma_{L_{\mathbf{r}}}(u_{\mathbf{r}}, v_{\mathbf{r}}) \cdot \sigma_{L_{\mathbf{a}}}(u_{\mathbf{a}}, v_{\mathbf{a}}).$$

Integration of these probabilities over the sieving region gives an estimate for the total yield of $(L_{\mathbf{r}}, L_{\mathbf{a}})$ -partial relations. An estimate for $\#V$ in the runtime of Coppersmith's variant is obtained by integrating the $\sigma_{L_{\mathbf{r}}}(u_{\mathbf{r}}, v_{\mathbf{r}})$ values over the sieving region. Similar integrations are used to compute candidate frequencies in Appendix B. A correction factor $6/\pi^2 \approx 0.608$ is applied to all results to account for the probability that a and b are coprime. The integrations were carried out using *Mathematica* and the *GNU Scientific Library*.

Actual Smoothness Tests. To get an impression of the accuracy of the above ρ and σ_1 -based estimates compared to the actual NFS yield, we tested $N_{\mathbf{r}}(a, b)$ and $N_{\mathbf{a}}(a, b)$ -values for smoothness for wide ranges of (a, b) pairs. Because it has never been doubted that the probability that $N_{\mathbf{r}}(a, b)$ and $N_{\mathbf{a}}(a, b)$ are smooth equals the product of the smoothness probabilities, we did not test that assumption.

We had no access to a sieve that allows the range of factor base sizes we intended to test, nor to hardware on which it would be able to run efficiently. Therefore we wrote a smoothness test that uses trial division up to 2^{30} combined with the elliptic curve factoring method (ECM). The choice 2^{30} was partially inspired by our wish not to miss any semi-smooth $N_{\mathbf{r}}(a, b)$ or $N_{\mathbf{a}}(a, b)$ -values that would, in theory, be found when using one of the parameter choices from [17].

The simplest approach would have been to subject each successive number to be tested to trial division followed, if necessary, by the ECM. To obtain slightly greater speed, and without having to deal with the imperfections (overlooking smooth values) and inconveniences (memory requirements, resieving or trial divisions to obtain the cofactor) of sieving, the trial divisions were organized in such a way that a large consecutive range of a 's could be handled reasonably efficiently, for a fixed b . For the algebraic norms this was achieved as follows (the rational norms are processed similarly). Let $[A_1, A_2]$ be a range of a -values to be processed. For all (prime, root) pairs (p, r) with $p < 2^{30}$ calculate the smallest $a_p \geq A_1$ such that $a_p \equiv br \pmod{p}$ (i.e., p divides $N_{\mathbf{a}}(a_p, b)$) and if $a_p \leq A_2$ insert the pair (p, a_p) in a heap that is ordered with respect to non-decreasing a_p values. Next, for $a = A_1, A_1 + 1, \dots, A_2$ in succession compute $c_a = N_{\mathbf{a}}(a, b)$, remove all elements with $a_p = a$ from the top of the heap, remove all corresponding factors p from c_a , and if $a_p + p \leq A_2$ insert $(p, a_p + p)$ in the heap. Note that this can be

seen as a variant of the ‘largish station’ design from [18]. The resulting c_a values have no factors $< 2^{30}$, are prime if $< 2^{60}$, and subjected to the ECM if composite. Due to the probabilistic nature of the ECM, factors between 2^{30} and the smoothness bound y_a (or y_r) may be overlooked. With proper ECM parameter settings and reasonably sized y_a (and y_r) this does not occur often. Furthermore, no relation relevant for the primary choice in [17] will be overlooked.

4 Traditional Extrapolation

In this section we sketch the traditional approach to estimate the difficulty of factoring a 1024-bit RSA modulus. Let R indicate a resource required for a factorization effort. For instance, R could indicate the computing time or it could be the factor base size, or the total matrix weight, or any other aspect of the factorization for which one wants to measure the cost or size.

For each resource R let $C_R(x)$ be a function that measures, asymptotically for $x \rightarrow \infty$ and in the relevant unit, how much of R is needed to factor x . For several resources a theoretical expression for this function is known. For instance, when R measures the total expected computing time, then

$$C_R(x) \approx L_x[1/3, (64/9)^{1/3}],$$

with $L_x[.]$ as in Section 3. If R measures the factor base size the constant $(64/9)^{1/3}$ in this expression would, in theory, be halved.

Assume that $R_{n'}$ units of some resource R are known to be required (or were used) to factor some RSA modulus n' . Then $\frac{C_R(n)}{C_R(n')} R_{n'}$ is used to estimate how much of R would be required (or feasible) for the factorization of RSA modulus n . In this type of estimate it is customary to ignore all $o(1)$ ’s, if they occur in C_R . Based on frequent observations this is not unreasonable if $\log(n')$ and $\log(n)$ are close. For large scale extrapolations, however, omitting the $o(1)$ ’s may be an over-simplification that might produce misleading results.

Furthermore, even if $\log(n')$ and $\log(n)$ are close, C_R -based extrapolation for resources R that are well understood in theory, may lead to results that have no practical value. As an example, for a 512-bit factorization, e.g. RSA-155, one would recommend a factor base size that is about 2.5 times larger than for a 462-bit factorization (as RSA-140). In practice, however, the entire concept of factor base size is obscured by the use of multiple large primes and special q ’s: it turned out that using the same factor base size did not lead to severe performance degradation.

This particular effect that not-even-nearly-optimal factor base sizes still lead to only slightly suboptimal performance is due to the behavior around the minimum of the runtime curve as a function of the factor base size: the runtime only gradually increases for factor base sizes that are much larger or somewhat smaller than the optimum. On the other hand, it increases sharply if the factor base size gets much too small (cf. [20]). This explains the potential dangers of $o(1)$ -less factor base size extrapolation: a suboptimal small choice, in the region

where the curve is relatively well behaved, for the factor base for n' may extrapolate to a factor base size for n in the steep region of the curve, thereby leading to a much larger total runtime for n than anticipated; see also Section 5, Table 2.

It is not uncommon to use $n' = \text{RSA-155}$ (a 512-bit number) as the basis for the extrapolation. In [11] the following parameters were proposed for 512-bit numbers (in the notation of Section 2), which is close to the values used for the factorization of RSA-155 (cf. [4]):

512-bit moduli: $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{24}$, sieving region of size $S = 1.6\text{E}16$ ($A = 9\text{E}9$, $B = 9\text{E}5$; we use ‘ $v\text{EW}$ ’ for ‘ $v \cdot 10^w$ ’). According to [17] the sieving step can be done in less than ten minutes on a US\$10K device.

Straightforward ($o(1)$ -less) extrapolation suggests that 768 and 1024-bit moduli would require smoothness bounds that are 75 and 2700 times larger and sieving regions that are 6000 and $7.5\text{E}6$ times larger, respectively: smoothness bounds approximately 2^{30} and 2^{35} and $S \approx 1\text{E}20$ and $S \approx 1.2\text{E}23$, respectively. As shown in [12] additional optimization arguments may enter into and further complicate the extrapolation. In [17] this leads to relatively small estimates for the smoothness bounds and relatively large sieving regions:

768-bit moduli: $y_{\mathbf{r}} = y_{\mathbf{a}} = 1.2\text{E}7$ ($< 2^{24}$), $S = 4.2\text{E}20$ ($A = 1.5\text{E}12$, $B = 1.5\text{E}8$). The sieving step can be done within 70 days on a US\$5K device.

1024-bit moduli: $y_{\mathbf{r}} = y_{\mathbf{a}} = 2.5\text{E}8$ ($< 2^{28}$), $S = 6\text{E}23$ ($A = 5.5\text{E}13$, $B = 5.5\text{E}9$). The sieving step takes a year on a US\$10M device.

Furthermore, the following is given in [17] and claimed to be an overestimate based on traditional extrapolation:

1024-bit moduli, but not using partial relations: $y_{\mathbf{r}} = y_{\mathbf{a}} = 1.5\text{E}10$ ($< 2^{34}$), $S = 6\text{E}23$. The sieving step takes a year on a US\$50M device.

5 Results

Let the notation be as in Section 2. In this section we present our ρ and σ_1 -based estimates for the yield of the NFS when applied to RSA-155 and RSA-768 with the parameters as suggested in [17] (and specified in Section 4) and to RSA-1024 for a wide variety of parameters, including those from [17]. Furthermore, we compare the estimates to the results of smoothness tests applied to numbers that would occur in an actual NFS factorization attempt. In Appendix B we give the corresponding estimates for RSA-1024 and the parameter choices from [18].

512-bit Moduli. Let $n = \text{RSA-155}$, $d = 5$, f as in [4], $s = 10800$, and $t = \exp(5.3)$. Application of our ρ and σ_1 -based estimates to $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{24}$, $z_{\mathbf{r}} = z_{\mathbf{a}} = 2^6 y_{\mathbf{r}} = 2^{30}$, $A = 9\text{E}9$, and $B = 9\text{E}5$ result in an estimated yield of $T(y_{\mathbf{r}}, y_{\mathbf{a}})/8.9 \approx 2.4\text{E}5$ **ff**s, $2.2\text{E}6$ **fp**s, $9.1\text{E}5$ **pf**s, and $8.1\text{E}6$ **pp**s. Because the parameter choice was intended for the use of more than a single large prime per norm, these results look acceptable: if more than one tenth of the matrix is filled

with \mathbf{ff} s, combinations of multi-prime partial relations will certainly fill in the rest.

With $y_r = 2^{29}$, $y_a = 2^{30}$, and $B = 4.0\text{E}4$ the same fraction of the matrix would be filled with \mathbf{ff} s for a sieving effort that is more than 470 times lower, but $T(y_r, y_a)$ would be 38.4 times larger, and sieving would have required more fast RAM than was available in 1999. Because $y_r = y_a = 2^{24}$ is much smaller than the choice that would minimize the sieving effort, extrapolation may result in very large sieving efforts, as mentioned in Section 4. See also Table 2 below.

768-bit Moduli. For $n = \text{RSA-768}$ we generated a fifth degree polynomial with $s \approx 26000$ and $t \approx \exp(5.3)$. To get $S = 4.2\text{E}20$, we use $B = 9\text{E}7$ and $A = sB$. With $y_r = y_a = 2^{24}$, $T(y_r, y_a) = 2.1\text{E}6$, and $z_r = z_a = 2^{10}y_r = 2^{34}$ we estimate a yield of fewer than 40 \mathbf{ff} s, 1200 \mathbf{fp} 's, 500 \mathbf{pf} s, and $2\text{E}4$ \mathbf{pp} 's. It is unlikely that this is feasible, unless a substantial effort is spent on finding multi-prime partial relations. With $y_r = 2^{29}$, $y_a = 2^{30}$, and the same sieving region, about $T(y_r, y_a)/16 \approx 5.2\text{E}6$ \mathbf{ff} s can be expected. With reasonable use of partial relations this may be feasible.

1024-bit Moduli. For $n = \text{RSA-1024}$ we considered degrees $d = 5, 6, 7, 8, 9$, each with corresponding integer m , d -th degree polynomial f , skewness ratio s , and correction factor t as specified in Appendix A. For each of these degrees and $S = 6\text{E}23$ the estimated yield figures are presented in the first two parts of Table 1, both for $y_r = y_a = 2^{28}$ and $y_r = y_a = 2^{34}$. Because the skewness ratio s depends on d , the height $B = \sqrt{S/(2s)}$ and width $2A = 2sB$ of the sieving region depend on d . In the last two parts the effect is given of doubling and quadrupling B , thereby increasing S (and the sieving effort) by a factor 4 and 16, respectively (since the skewness ratio s is kept invariant). We used $z_r = z_a = 2^j y_r$ for $j \in \{8, 12, 16\}$ and indicate the expected \mathbf{fp} , \mathbf{pf} , and \mathbf{pp} yield by \mathbf{fp}_j , \mathbf{pf}_j , and \mathbf{pp}_j , respectively. Note that [17] does not use partial relations for $y_r = y_a = 2^{34}$.

Table 1. Estimated yields for smoothness bounds from [17].

d	s	B	\mathbf{ff}	\mathbf{fp}_8	\mathbf{pf}_8	\mathbf{pp}_8	\mathbf{fp}_{12}	\mathbf{pf}_{12}	\mathbf{pp}_{12}	\mathbf{fp}_{16}	\mathbf{pf}_{16}	\mathbf{pp}_{16}
$y_r = y_a = 2^{28}$, $T(y_r, y_a) \approx 2.9\text{E}7$, $S = 6\text{E}23$, sieving effort $3.6\text{E}24$												
5	87281.9	1.9E9	22	4.7E2	2.3E2	5.1E3	9.2E2	4.3E2	1.8E4	1.6E3	6.9E2	5.1E4
6	458.9	2.6E10	74	1.7E3	6.3E2	1.4E4	3.3E3	1.1E3	5.0E4	5.8E3	1.8E3	1.4E5
7	40.9	8.6E10	1.5E2	3.6E3	1.0E3	2.4E4	6.9E3	1.8E3	8.1E4	1.2E4	2.8E3	2.2E5
8	107.3	5.3E10	34	8.2E2	1.8E2	4.5E3	1.6E3	3.2E2	1.5E4	2.8E3	4.8E2	4.0E4
9	8.5	1.9E11	3	69	14	2.5E2	1.3E2	24	8.2E2	1.8E2	37	2.2E3
$y_r = y_a = 2^{34}$, $T(y_r, y_a) \approx 1.5\text{E}9$, $S = 6\text{E}23$, sieving effort $3.8\text{E}24$												
5	87281.9	1.9E9	9.1E6	1.1E8	5.6E7	6.9E8	2.0E8	9.5E7	2.1E9	3.3E8	1.5E8	5.2E9
6	458.9	2.6E10	2.1E7	2.8E8	1.0E8	1.4E9	5.1E8	1.7E8	4.1E9	8.2E8	2.6E8	1.0E10
7	40.9	8.6E10	3.1E7	4.3E8	1.2E8	1.7E9	7.7E8	2.0E8	5.0E9	1.3E9	2.9E8	1.2E10
8	107.3	5.3E10	6.8E6	1.0E8	2.2E7	3.3E8	1.9E8	3.6E7	9.9E8	3.1E8	5.2E7	2.4E9
9	8.5	1.9E11	5.3E5	8.5E6	1.5E6	2.5E7	1.6E7	2.5E6	7.3E7	2.6E7	3.6E6	1.8E8
$y_r = y_a = 2^{34}$, $T(y_r, y_a) \approx 1.5\text{E}9$, $S = 2.4\text{E}24$, sieving effort $1.5\text{E}25$												
5	87281.9	3.7E9	1.9E7	2.4E8	1.2E8	1.5E9	4.4E8	2.0E8	4.6E9	7.1E8	3.1E8	1.1E10
6	458.9	5.1E10	4.0E7	5.6E8	2.0E8	2.7E9	1.0E9	3.3E8	8.1E9	1.6E9	5.0E8	2.0E10
7	40.9	1.7E11	5.2E7	7.4E8	2.0E8	2.9E9	1.3E9	3.3E8	8.7E9	2.2E9	5.0E8	2.1E10
$y_r = y_a = 2^{34}$, $T(y_r, y_a) \approx 1.5\text{E}9$, $S = 9.8\text{E}24$, sieving effort $6.1\text{E}25$												
5	87281.9	7.4E9	4.1E7	5.3E8	2.5E8	3.3E9	9.5E8	4.3E8	1.0E10	1.5E9	6.6E8	2.5E10
6	458.9	1.0E11	7.5E7	1.0E9	3.7E8	5.2E9	2.0E9	6.3E8	1.6E10	3.2E9	9.5E8	3.9E10
7	40.9	3.4E11	8.6E7	1.3E9	3.4E8	5.0E9	2.3E9	5.7E8	1.5E10	3.8E9	8.4E8	3.7E10

It follows from Table 1 that unless multi-prime partial relations are collected on a much wider scale than customary or practical, the choice $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{28}$, and thus the smaller choice $y_{\mathbf{r}} = y_{\mathbf{a}} = 2.5\text{E}8$ from [17], looks infeasible. Also the choice $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, and therefore the choice $y_{\mathbf{r}} = y_{\mathbf{a}} = 1.5\text{E}10$ from [17], is infeasible if, as suggested in [17], partial relations are not used and if a sieving region size S as proposed in [17] is used. To get the choice $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$ to work without partial relations, our estimates suggest that $d = 6$ with $B \approx 2.9\text{E}12$ (corresponding to $S \approx 8\text{E}27$) would suffice. This would, however, be about 13000 times more expensive than the estimate from [17]: the initial $2.6\text{E}10$ b -values produce about $T(y_{\mathbf{r}}, y_{\mathbf{a}})/72$ \mathbf{ff} s, but the performance deteriorates for larger b 's so that much more than 72 times the initial effort is needed to find $T(y_{\mathbf{r}}, y_{\mathbf{a}})$ \mathbf{ff} s. For $d = 5$ or 7 it would be 1.1 or 3.5 times more expensive, respectively.

Using partial relations is probably a more efficient way to get $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$ to work, as suggested by the last two parts of Table 1. Since there are no adequate methods yet to predict if the partial relation yield as listed, in practice augmented with partial relations with 3 or more large primes, would suffice or not, we cannot make any definite statements on the resulting cost, the practical merit of the cost estimate from [17], or the semi-smoothness bound that would be required. Note that the performance of $d = 6, 7$ deteriorates faster than for $d = 5$, as expected.

In Table 2 the effect of low smoothness bounds is illustrated. The total expected sieving effort to find $T(y_{\mathbf{r}}, y_{\mathbf{a}})/32$ \mathbf{ff} s is listed for $d = 6$, $y_{\mathbf{r}} = 2^j$ with $j = 28, 29, \dots, 51$ and $y_{\mathbf{a}} = 2y_{\mathbf{r}}$. The optimum $9.3\text{E}20$ is achieved at $j = 47$. When j gets smaller the effort at first increases slowly and gradually, but around $j = 39$ the effort grows faster than the smoothness bounds shrink, and for smaller j the performance deteriorates rapidly.

Table 2. Sieving effort to find $T(2^j, 2^{j+1})/32$ \mathbf{ff} s for $d = 6$.

j	effort	j	effort	j	effort	j	effort	j	effort	j	effort
28	1.5E36	32	5.6E26	36	1.7E23	40	4.8E21	44	1.2E21	48	9.6E20
29	4.7E32	33	3.7E25	37	5.2E22	41	2.9E21	45	1.0E21	49	1.0E21
30	1.4E30	34	4.2E24	38	2.0E22	42	2.0E21	46	9.4E20	50	1.2E21
31	1.7E28	35	7.2E23	39	9.1E21	43	1.5E21	47	9.3E20	51	1.4E21

We now vary d and $i_{\mathbf{r}}, i_{\mathbf{a}} \in \{25, 26, \dots, 50\}$ and minimize the sieving effort to find $T(2^{i_{\mathbf{r}}}, 2^{i_{\mathbf{a}}})/c$ \mathbf{ff} s, for various c 's. The resulting sieving efforts with corresponding optimal smoothness bounds are listed in Table 3. It can be seen that both effort and smoothness bounds decrease with increasing c . This effect is stronger for larger d . Overall, $d = 7$ is the best choice, with $d = 6$ better than $d = 8$ for small c but vice versa for larger ones. For non-optimal smoothness bounds, however, $d = 7$ may not be the best choice, as illustrated in Table 1.

Actual Smoothness Tests for RSA-1024. The accuracy of our ρ and σ_1 -based estimates as derived for $n = \text{RSA-1024}$ was tested by applying smoothness tests (as explained in Section 3) to $N_{\mathbf{r}}(a, b)$ and $N_{\mathbf{a}}(a, b)$ -values for wide ranges of (a, b) -pairs with coprime a and b and degrees and parameters as in Appendix A. More than 100 billion values have been tested for degrees 6 and 7. No major

Table 3. Minimal sieving efforts to find $T(2^i, 2^j)/c$ \mathbf{ff} s.

d	$c = 1$			$c = 8$			$c = 16$			$c = 32$			$c = 64$			$c = 128$		
	i_r, i_a	effort		i_r, i_a	effort		i_r, i_a	effort		i_r, i_a	effort		i_r, i_a	effort		i_r, i_a	effort	
6	48,49	1.6E23		47,48	7.2E21		47,48	2.6E21		47,48	9.2E20		47,48	3.3E20		46,47	1.2E20	
7	47,49	9.4E22		47,49	3.5E21		46,48	1.1E21		46,47	3.5E20		45,47	1.1E20		45,46	3.6E19	
8	48,50	3.7E23		47,49	1.0E22		46,48	3.0E21		46,48	8.7E20		45,47	2.5E20		45,47	7.5E19	

Table 4. Actual and estimated number of $(2^i, 2^j, 1)$ -semi-smooth $N_r(a, b)$'s for $d = 6$.

j	i						
	24	25	26	27	28	29	30
24	2.4E3(2.7E3)						
25	4.9E3(5.5E3)	6.3E3(7.0E3)					
26	7.7E3(8.6E3)	1.2E4(1.4E4)	1.5E4(1.7E4)				
27	1.1E4(1.2E4)	1.9E4(2.1E4)	2.8E4(3.1E4)	3.4E4(3.7E4)			
28	1.4E4(1.6E4)	2.6E4(2.9E4)	4.3E4(4.7E4)	6.1E4(6.7E4)	7.1E4(7.7E4)		
29	1.8E4(2.0E4)	3.4E4(3.7E4)	5.8E4(6.4E4)	8.9E4(9.8E4)	1.2E5(1.3E5)	1.4E5(1.5E5)	
30	2.2E4(2.4E4)	4.2E4(4.7E4)	7.5E5(8.2E4)	1.2E5(1.3E5)	1.8E5(1.9E5)	2.3E5(2.5E5)	2.5E5(2.7E5)
31	2.6E4(2.9E4)	5.1E4(5.7E4)	9.3E4(1.0E5)	1.5E5(1.7E5)	2.4E5(2.6E5)	3.3E5(3.6E5)	3.8E5(4.1E5)
32	3.1E4(3.4E4)	6.1E4(6.8E4)	1.1E5(1.2E5)	1.9E5(2.1E5)	3.0E5(3.2E5)	4.3E5(4.7E5)	5.1E5(5.5E5)
33	3.6E4(4.0E4)	7.2E4(8.0E4)	1.3E5(1.5E5)	2.3E5(2.5E5)	3.7E5(4.0E5)	5.5E5(5.9E5)	6.5E5(7.1E5)
34	4.2E4(4.7E4)	8.4E4(9.3E4)	1.6E5(1.7E5)	2.7E5(3.0E5)	4.4E5(4.8E5)	6.7E5(7.2E5)	8.0E5(8.7E5)
35	4.8E4(5.4E4)	9.7E4(1.1E5)	1.8E5(2.0E5)	3.2E5(3.5E5)	5.2E5(5.6E5)	8.0E5(8.6E5)	9.7E5(1.0E6)
36	5.5E4(6.1E4)	1.1E5(1.2E5)	2.1E5(2.3E5)	3.6E5(4.0E5)	6.0E5(6.5E5)	9.3E5(1.0E6)	1.1E6(1.2E6)
37	6.3E4(7.0E4)	1.3E5(1.4E5)	2.4E5(2.6E5)	4.2E5(4.6E5)	6.9E9(7.5E5)	1.1E6(1.2E6)	1.3E6(1.4E6)
38	7.1E4(7.9E4)	1.4E5(1.6E5)	2.7E5(3.0E5)	4.7E5(5.2E5)	7.8E5(8.5E5)	1.2E6(1.3E6)	1.5E6(1.6E6)
39	8.1E4(9.0E4)	1.6E5(1.8E5)	3.0E5(3.3E5)	5.3E5(5.8E5)	8.9E5(9.7E5)	1.4E6(1.5E6)	1.7E6(1.9E6)
40	9.1E4(1.0E5)	1.8E5(2.0E5)	3.4E5(3.8E5)	6.0E5(6.6E5)	1.0E6(1.1E6)	1.6E6(1.7E6)	1.9E6(2.1E6)

surprises or unexpected anomalies were detected. Thus, although it may be too early to have complete confidence in the ρ and σ_1 -based estimates, there is neither any reason to dismiss them.

For $d = 6$ this is illustrated in Tables 4, 5, and 6. Tables 4 and 5 contain the accumulated results of smoothness tests for $N_r(a, b)$ and $N_a(a, b)$ -values, respectively, for more than 100 billion coprime (a, b) pairs and 176 different b values ranging from 2^9 to 2^{31} . They list the number of $(2^i, 2^j, 1)$ -semi-smooth $N_r(a, b)$ and $N_a(a, b)$ -values (for i, j ranges as specified in the tables) that were found using trial division up to 2^{30} , followed by the $(\rho + \sigma_1)$ -based estimate between parentheses. Table 6 contains the accumulated results of more expensive smoothness tests for $N_a(a, b)$ -values for 5.6 million coprime (a, b) pairs and 13 different b -values ranging from 2^{14} to 2^{26} . For $34 \leq j \leq 40$ and $31 \leq i \leq j$ it lists the number of $(2^i, 2^j, 1)$ -semi-smooth $N_a(a, b)$ -values, found using trial division up to 2^{30} followed by ECM, again followed by the $(\rho + \sigma_1)$ -based estimate between parentheses. The fact that the estimated value is systematically somewhat higher than the actual value can be attributed to the fact that the estimated values average over all positive numbers less than some bound, whereas most values that are actually tested are close to the bound. This is partly offset by the use of asymptotic smoothness probabilities, which are somewhat smaller than the concrete probabilities (e.g., for $\rho(u_r)$ the correction term is roughly $+0.423\rho(v_r - 1)/\log N_r(a, b)$; cf. [1]).

For $d = 7$ we found comparable results. Because of the asymptotic nature of the estimates, it may be expected that they become even more accurate for the larger b 's that may occur in practice (cf. Table 1).

Table 5. Actual and estimated number of $(2^i, 2^j, 1)$ -semi-smooth $N_{\mathbf{a}}(a, b)$'s for $d = 6$.

j	i						
	24	25	26	27	28	29	30
28	0(0.15)	0(0.41)	0(0.96)	0(1.85)	0(2.53)		
29	0(0.19)	1(0.54)	1(1.36)	1(2.94)	1(5.32)	1(7.01)	
30	0(0.23)	1(0.69)	1(1.80)	1(4.14)	1(8.34)	5(14.18)	10(16.87)
31	0(0.29)	1(0.86)	2(2.28)	2(5.45)	5(11.63)	17(21.94)	24(28.52)
32	1(0.34)	2(1.04)	3(2.81)	3(6.88)	8(15.21)	27(30.34)	40(41.10)
33	1(0.41)	2(1.24)	5(3.40)	5(8.45)	12(19.11)	39(39.44)	58(54.70)
34	1(0.48)	2(1.47)	5(4.05)	5(10.17)	15(23.36)	49(49.31)	70(69.41)
35	1(0.56)	2(1.72)	6(4.76)	7(12.05)	21(28.00)	60(60.01)	82(85.33)
36	1(0.65)	2(2.00)	7(5.55)	10(14.12)	27(33.05)	71(71.63)	97(102.57)
37	1(0.75)	2(2.30)	8(6.42)	11(16.39)	31(38.58)	82(84.26)	111(121.26)
38	2(0.86)	3(2.65)	9(7.38)	12(18.88)	36(44.61)	95(97.98)	132(141.52)
39	2(0.99)	3(3.03)	10(8.45)	14(21.62)	41(51.20)	106(112.90)	148(163.51)
40	2(1.13)	3(3.46)	11(9.62)	19(24.63)	47(58.41)	115(129.13)	163(187.36)

Table 6. Actual and estimated number of $(2^i, 2^j, 1)$ -semi-smooth $N_{\mathbf{a}}(a, b)$'s for $d = 6$.

j	i									
	31	32	33	34	35	36	37	38	39	40
34	0(0.30)	0(0.49)	0(0.70)	0(0.82)						
35	0(0.39)	0(0.66)	0(1.03)	1(1.41)	1(1.62)					
36	0(0.48)	0(0.85)	0(1.38)	1(2.05)	1(2.73)	1(3.08)				
37	0(0.58)	0(1.05)	0(1.75)	2(2.72)	2(3.90)	2(5.03)	2(5.60)			
38	0(0.69)	0(1.26)	0(2.15)	2(3.44)	3(5.14)	4(7.11)	5(8.95)	5(9.84)		
39	1(0.81)	1(1.49)	1(2.58)	3(4.21)	4(6.46)	8(9.30)	9(12.48)	12(15.36)	13(16.72)	
40	1(0.93)	1(1.74)	1(3.04)	4(5.02)	6(7.86)	12(11.62)	15(16.20)	18(21.16)	21(25.51)	23(27.52)

6 More or Better Polynomials?

Estimating the Performance of Coppersmith's Variant. We estimated the yield and performance of Coppersmith's multi-polynomial version of the NFS by assuming that for any degree d we can find a set G of any reasonable cardinality consisting of degree d polynomials with a shared root m modulo n and with skewness ratios and correction factors comparable to those in Appendix A. Table 7 lists some estimates for $d = 6, 7$ and $\#G = 6$ that can be compared to the estimates in Table 1. The dimension of the matrix increases $7/2$ -fold and the yield improves by a factor 6. The **fp** and **pp** yield increase may not be that effective, since large primes match only if they occur in the norm of the same polynomial. The relation collection effort changes from sieving effort $3.8\text{E}24$ to sieving effort $1.9\text{E}24$ plus a number of semi-smoothness tests (indicated by 'ECM effort') involving a constant of proportionality E measuring the relative performance compared to sieving.

The practical implications are as yet unclear. For current implementations E would be too large to make the multi-polynomial version competitive, but an entirely different picture may emerge for dedicated non-sieving hardware smoothness tests. Also, our choices $d = 6, 7$ and $\#G = 6$ were not meant to optimize anything, they are just for illustrative purposes to facilitate comparison with the regular NFS data in Table 1. Clearly, this subject deserves further study.

The Effect of Much Better Polynomials. In an actual factorization attempt considerably more time would be spent to find good polynomials. So, in

Table 7. Estimated yields for smoothness bounds from [17] with 6 polynomials.

$y_r = y_a = 2^{34}$, goal $\approx 5.3\text{E}9$, $S = 6\text{E}23$, sieving effort $1.9\text{E}24$											
d	s	B	ff	fp_8	pf_8	pp_8	ECM effort	fp_{12}	pf_{12}	pp_{12}	ECM effort
6	458.9	2.6E10	1.3E8	1.7E9	6.2E8	8.2E9	$E5.6\text{E}20$	3.0E9	1.0E9	2.5E10	$E8.7\text{E}20$
7	40.9	8.6E10	1.8E8	2.6E9	7.1E8	9.9E9	$E3.6\text{E}21$	4.6E9	1.2E9	3.0E10	$E5.4\text{E}21$

Table 8. Estimated yields for smoothness bounds from [17] with correction factor t^3 .

d	s	B	ff	fp_8	pf_8	pp_8	fp_{12}	pf_{12}	pp_{12}	fp_{16}	pf_{16}	pp_{16}
$y_r = y_a = 2^{28}$, $T(y_r, y_a) \approx 2.9\text{E}7$, $S = 6\text{E}23$, sieving effort $3.6\text{E}24$												
6	458.9	2.6E10	2.6E2	5.8E3	2.2E3	4.9E4	1.1E4	4.0E3	1.7E5	2.0E4	6.3E3	4.7E5
7	40.9	8.6E10	6.8E2	1.5E4	4.6E3	1.0E5	2.9E4	8.0E3	3.5E5	5.1E4	1.2E4	9.5E5
$y_r = y_a = 2^{34}$, $T(y_r, y_a) \approx 1.5\text{E}9$, $S = 6\text{E}23$, sieving effort $3.8\text{E}24$												
6	458.9	2.6E10	5.7E7	7.2E8	2.8E8	3.5E9	1.3E9	4.9E8	1.1E10	2.1E9	7.3E8	2.6E10
7	40.9	8.6E10	9.9E7	1.3E9	3.9E8	5.1E9	2.4E9	6.4E8	1.5E10	3.9E9	9.4E8	3.7E10
$y_r = y_a = 2^{34}$, $T(y_r, y_a) \approx 1.5\text{E}9$, $S = 2.4\text{E}24$, sieving effort $1.5\text{E}25$												
6	458.9	5.1E10	1.1E8	1.4E9	5.3E8	6.9E9	2.5E9	8.9E8	2.1E10	4.1E9	1.3E9	5.1E10
7	40.9	1.7E11	1.7E8	2.3E9	6.6E8	9.1E9	4.2E9	1.1E9	2.7E10	6.8E9	1.6E9	6.5E10
$y_r = y_a = 2^{34}$, $T(y_r, y_a) \approx 1.5\text{E}9$, $S = 9.8\text{E}24$, sieving effort $6.1\text{E}25$												
6	458.9	1.0E11	2.0E8	2.8E9	1.0E9	1.4E10	4.9E9	1.7E9	4.1E10	8.0E9	2.6E9	1.0E11
7	40.9	3.4E11	2.8E8	4.0E9	1.1E9	1.6E10	7.3E9	1.9E9	4.8E10	1.2E10	2.8E9	1.2E11

practice, we may expect correction factors t that are larger than the ones given in Appendix A for polynomials which may have smaller coefficients. An example of such a polynomial is given in Appendix B. This effect can be approximated by applying our estimates to the same f and m values but with incorrect (too large) correction factors t . In Table 8 the results are given if t is replaced by t^3 for $d = 6, 7$, with parameters as in Table 1 (i.e., mostly as in [17]). With the current state of the art of polynomial selection methods it is unlikely that such large correction factors can be found in practice. Thus, the figures in Table 8 are probably too optimistic. Compared to Table 1 the yield improves by a factor about 3: a relatively small effect that does not have an impact on the observations made in Section 5 about $y_r = y_a = 2^{28}$ and $y_r = y_a = 2^{34}$. For $d = 6$ and $y_r = y_a = 2^{34}$ not using partial relations (and correction factor t^3) would require $B = 9.4\text{E}11$ with corresponding $S = 8.2\text{E}26$. This is about 1300 times more expensive than the estimate from [17]. We conclude that our limited polynomial search did not lead to overly poor estimates.

7 Conclusion

We applied numerical methods to estimate the yield of the NFS when applied to the 1024-bit RSA modulus RSA-1024, and tested the accuracy of our results using actual smoothness tests. Our methods and results were taken into account in the updated version [18] of the draft version of TWIRL [17] and are presented in Appendix B. Accurate estimates of the difficulty of factoring 1024-bit RSA moduli require a better understanding of the large prime matching behavior than is available today. Continued large factorization efforts may prove helpful.

Our results suggest that effective smoothness bounds for RSA-1024 are larger than the ones proposed in [17]. Larger smoothness bounds stress the importance

of the alternative cost measure proposed in [2] and of approaches to smoothness testing that avoid sieving and storage of the complete factor bases. TWINKLE and TWIRL (cf. [19], [18]) both require processing elements or storage for essentially the complete factor bases and time for the sieving. Such designs may eventually be surpassed by, say, a carefully designed ECM-based smoothness test as proposed in [2], because the latter allows a better trade-off between space and time. This does not disqualify TWIRL for the sizes proposed in [18], but indicates that in the long term the approach from [2] may be more promising.

Acknowledgment

We thank Mike Szydlo for useful discussions, and for sharing his observations about [17]. We are grateful to Thorsten Kleinjung and Jens Franke for their polynomial selection program and subsequent discussions.

References

1. E. Bach, R. Peralta, *Asymptotic semi-smoothness probabilities*, University of Wisconsin, Technical report #1115, October 1992
2. D.J. Bernstein, *Circuits for integer factorization: a proposal*, manuscript, November 2001; available at http://cr.yp.to/papers.html#nfs_circuit
3. E.R. Canfield, P. Erdős, C. Pomerance, *On a problem of Oppenheim concerning "Factorisatio Numerorum"*, J. Number Theory **17** (1983) 1–28
4. S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgomery, B. Murphy, H.J.J. te Riele, et al., *Factorization of a 512-bit RSA modulus*, Proceedings Eurocrypt 2000, LNCS 1807, Springer-Verlag 2000, 1–17
5. D. Coppersmith, *Modifications to the number field sieve*, Journal of Cryptology **6** (1993) 169–180
6. R. Crandall, C. Pomerance, *Prime numbers*, Springer-Verlag, 2001
7. N.G. De Bruijn, *On the number of positive integers $\leq x$ and free of prime factors $> y$, II*, Indag. Math. **38** (1966) 239–247
8. International Technology Roadmap for Semiconductors 2002 Update, <http://public.itrs.net/>
9. R. Lambert, *Computational aspects of discrete logarithms*, Ph.D. thesis, University of Waterloo, 1996.
10. A.K. Lenstra, H.W. Lenstra, Jr., (eds.), *The development of the number field sieve*, Lecture Notes in Math. **1554**, Springer-Verlag 1993
11. A.K. Lenstra, A. Shamir, *Analysis and optimization of the TWINKLE factoring device*, Proceedings Eurocrypt 2000, LNCS 1807, Springer-Verlag 2000, 35–52
12. A.K. Lenstra, A. Shamir, J. Tomlinson, E. Tromer, *Analysis of Bernstein's factorization circuit*, Proceedings Asiacrypt 2002, LNCS 2501, Springer-Verlag 2002, 1–26
13. P.L. Montgomery, B. Murphy, *Improved polynomial selection for the number field sieve*, extended abstract for the conference on the mathematics of public-key cryptography, June 13–17, 1999, The Fields institute, Toronto, Ontario, Canada
14. B. Murphy, *Modelling the yield of the number field sieve polynomials*, Proceedings ANTS-III, LNCS 1423, Springer-Verlag, 1998, 137–151

15. B. Murphy, *Polynomial selection for the number field sieve integer factorisation algorithm*, PhD thesis, The Australian National University, July 1999
16. RSA Challenge Administrator, see <http://www.rsasecurity.com/rsalabs/challenges/factoring/index.html>
17. A. Shamir, E. Tromer, *Factoring large numbers with the TWIRL device (preliminary draft)*, February 4, 2003; available at www.wisdom.weizmann.ac.il/~tromer/papers/twirl-20030208.ps.gz
18. A. Shamir, E. Tromer, *Factoring large numbers with the TWIRL device*, Proceedings Crypto 2003, LNCS 2729, Springer-Verlag 2003, 1–26
19. A. Shamir, *Factoring large numbers with the TWINKLE device*, Proceedings CHES'99, LNCS 1717, Springer-Verlag, 1999
20. R.D. Silverman, *Optimal parameterization of SNFS*, Manuscript, 2002

A Polynomials for RSA-1024

Let the notation be as in Section 2. RSA-1024 = 135...563 is a 1024-bit number whose 309 decimal digits can be found in [16]. For $d = 5, 6, 7, 8, 9$ we present the value of m , the skewness ratio s , the correction factor t , and the d -th degree polynomial f . For all d we have that $f(m) = \text{RSA-1024}$ and the number of free relations behaves as estimated in Section 2.

- $d = 5$: $m = 40166061499405767761275922505205845319620673223962394269848$,
 $s = 87281.9$, $t = \exp(4.71)$,

$$f(X) = 1291966090228800X^5 - 640923572655549773652421X^4$$

$$+ 22084609569698872827347541432045436154518749958885X^3$$

$$+ 395968894120701874630226095753546547718334332711719805X^2$$

$$- 96965973957066386285836042292532199420340774279358321957826X$$

$$- 4149238485198657863882627412883817567549615187136520422680871493.$$
- $d = 6$: $m = 6290428606355899027255723320027391715970345088070$, $s = 458.857$, $t = \exp(3.10)$,

$$f(X) = 2180047385355840X^6 - 3142872579455569636X^5$$

$$- 1254155662796860036208992514969847001569768X^4$$

$$- 12346184596682129311885354974311793670338999X^3$$

$$+ 326853630498301587526877377811152784944999520522X^2$$

$$+ 4609395911122979440239635705733809071478223546768X$$

$$- 11074692768758259967955017581674706364925519996590997.$$
- $d = 7$: $m = 103900297567818360319524643906916425458585$, $s = 40.9082$, $t = \exp(3.66)$,

$$f(X) = 1033308066924956844000X^7 - 160755011543490353038479X^6$$

$$- 195303627236151056576676296300427751X^5$$

$$- 67322997660970472962322331424620518857X^4$$

$$+ 852886687422682194441338494667584979283X^3$$

$$+ 122261247387346205137507554160155213223449X^2$$

$$- 941042262598628457425892609296624845278218X$$

$$- 38806712095590448575304126518627120637325432.$$
- $d = 8$: $m = 1364850538695913738402818687041215458$, $s = 107.255$, $t = \exp(5.13)$,

$$f(X) = 11216738509080904800X^8 + 4126963962861489385859X^7$$

$$- 1175791917822439782941507504635X^6$$

$$+ 2996639999067533888196133035298645X^5$$

$$+ 208240147656019048048262524877102283X^4$$

$$- 27357702926139861867857609251152887873X^3$$

$$- 3424834099100207742896726960114709926535X^2$$

$$- 12957538712647811491436510238283188219229X$$

$$+ 8733287829967486818441309661955398847347705.$$

$$\begin{aligned}
d = 9: \quad & m = 1310717071544062886859477360545488, \quad s = 8.51584, \quad t = \exp(3.89), \\
& f(X) = 11829510000X^9 - 323042712742X^8 - 2296009166444361125150144310X^7 \\
& \quad - 17667833832765445702215975840307X^6 \\
& \quad + 104750984243461509795139799847908X^5 \\
& \quad + 684082899341824778960200186325064X^4 \\
& \quad - 8558486132848151826178414424938636X^3 \\
& \quad + 32301718781994667946436083991144874X^2 \\
& \quad - 42118837302218928303637260451515638X \\
& \quad - 1293558869408225281960437545569172565.
\end{aligned}$$

B The Parameter Settings from [18]

This appendix provides analysis of the NFS parameters used in the revised TWIRL design [18]. It follows the approach of Section 3, extended to produce estimates for the frequency of intermediate candidates.

Polynomials. We used the NFS polynomial selection program of Jens Franke and Thorsten Kleinjung, which contains several improvements on the strategy of [13][14][15] which was used to obtain the polynomials of Section 3 and Appendix A. We employed several Pentium 1.7GHz computers, for a total CPU time of about 20 days. However, most of this time was spent on experimentation with search parameters; the conclusions can be reused for other composites, so future experiments would require just a few hours. We observe that with this polynomial selection program there is a lot of flexibility in the search parameters: at a small cost in yield, one can obtain polynomials with much larger or much smaller skew, trade root properties for size properties, etc. Appendix B.2 of [18] gives the best polynomial we found for RSA-1024, which is as follows:

$$\begin{aligned}
d = 5: \quad & m = 2626198687912508027781823689041581872398105941296246738850076103682306196740 \\
& \quad 55292506154513387298663560887146183854975198160502278243245067074820593711054723850 \\
& \quad 5700273957561400114202031348071179037320617188128273668251667043465012822281608387 \\
& \quad 169409282469138311259520392769843104985793744494821437272961970486, \\
& \quad s = 1991935.4, \quad t = \exp(6.33), \\
& \quad f(X) = 1719304894236345143401011418080X^5 \\
& \quad \quad - 699197348886605861074074186043634471X^4 \\
& \quad \quad + 27086030483569532894050974257851346649521314X^3 \\
& \quad \quad + 46937584052668574502886791835536552277410242359042X^2 \\
& \quad \quad - 101070294842572111371781458850696845877706899545394501384X \\
& \quad \quad - 22666915939490940578617524677045371189128909899716560398434136, \\
& \quad g(X) = 93877230837026306984571367477027X \\
& \quad \quad - 37934895496425027513691045755639637174211483324451628365.
\end{aligned}$$

Here the rational-side polynomial g is non-monic; thus we redefine $N_{\mathbf{r}}(a, b) = |b \cdot g(a/b)|$. Table 9 estimates the yield of this polynomial using the parameter sets from [17] that were considered in Section 5. A comparison with Table 1 shows that this polynomial has much higher yield; indeed, both its size properties and its root properties are better (cf. [15]). Throughout this appendix we shall use this polynomial, except where noted otherwise.

Note that Section 5 gives strong indication that $d = 5$ is suboptimal, but the program we used is limited to $d = 5$. One can expect that an adaptation of the improved algorithm to $d = 6$ or $d = 7$ will yield even better results. In this light, the parameters of [18] merely imply an upper bound on cost; further improvement is likely to be possible.

Yield. To increase yield, [18] uses higher smoothness bounds than [17]: $y_{\mathbf{r}} = 3.5\text{E}9$, $y_{\mathbf{a}} = 2.6\text{E}10$, $z_{\mathbf{r}} = 4.0\text{E}11$, $z_{\mathbf{a}} = 6.0\text{E}11$. This has a dramatic effect,

Table 9. Estimated yields with [18]’s RSA-1024 polynomial and [17]’s parameters.

d	B	$\mathfrak{f}\mathfrak{f}$	$\mathfrak{f}p_8$	$\mathfrak{p}\mathfrak{f}_8$	$\mathfrak{p}p_8$	$\mathfrak{f}p_{12}$	$\mathfrak{p}\mathfrak{f}_{12}$	$\mathfrak{p}p_{12}$	$\mathfrak{f}p_{16}$	$\mathfrak{p}\mathfrak{f}_{16}$	$\mathfrak{p}p_{16}$
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{28}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 2.9\text{E}7$, $S = 6\text{E}23$, sieving effort $3.6\text{E}24$											
5	3.88E8	9.9E2	2.0E4	9.7E3	2.0E5	3.8E4	1.8E4	6.8E5	6.6E4	2.8E4	1.9E6
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 1.5\text{E}9$, $S = 6\text{E}23$, sieving effort $3.8\text{E}24$											
5	3.88E8	1.8E8	2.1E9	1.0E9	1.2E10	3.7E9	1.7E9	3.5E10	5.9E9	2.6E9	8.6E10
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 1.5\text{E}9$, $S = 2.4\text{E}24$, sieving effort $1.5\text{E}25$											
5	3.88E8	3.8E8	4.5E9	2.2E9	2.5E10	8.1E9	3.7E9	7.7E10	1.3E10	5.6E9	1.9E11
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 1.5\text{E}9$, $S = 9.8\text{E}24$, sieving effort $6.1\text{E}25$											
5	3.88E8	8.2E8	9.9E9	4.7E9	5.7E10	1.8E10	8.0E9	1.7E11	2.9E10	1.2E10	4.2E11

Table 10. RSA-1024 parameters and estimates for [18].

$y_{\mathbf{r}} = 3.5\text{E}9$, $y_{\mathbf{a}} = 2.6\text{E}10$, $z_{\mathbf{r}} = 4.0\text{E}11$, $z_{\mathbf{a}} = 6.0\text{E}11$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 1.3\text{E}9$, $S = 3.0\text{E}23$ $d = 5$, $s = 1991935.4$, $B = 2.7\text{E}8$									
yield of $(L_{\mathbf{a}}, L_{\mathbf{r}})$ -partial relations									
(0, 0)	(0, 1)	(0, 2)	(1, 0)	(1, 1)	(1, 2)	(2, 0)	(2, 1)	(2, 2)	Total
5.6E7	3.0E8	6.7E8	3.1E8	1.7E9	3.8E9	6.6E8	3.5E9	7.9E9	1.9E10
#PRS	#PBS	#PPT	#RCF	#RSS	#ACF	avg($N_{\mathbf{r}}(a, b)$)		avg($N_{\mathbf{a}}(a, b)$)	
1.1E20	5.0E12	6.2E10	4.9E10	3.4E10	2.7E10	5.2E63		3.1E103	

suggesting that the choice from [17] indeed resides on the steep region of the run-time curve (cf. Section 4). Also, the number of allowed large primes is increased to $\ell_{\mathbf{r}} = \ell_{\mathbf{a}} = 2$. Conversely, the sieving region size is reduced to $S = 3.0\text{E}23$. Table 10 gives the corresponding estimates of yield, as well as the number of intermediate candidates (see below). Note that [18] uses different notation: there R , H , $B_{\mathbf{R}}$ and $B_{\mathbf{A}}$ stand for our $2A$, B , $y_{\mathbf{r}}$ and $y_{\mathbf{a}}$, respectively.

Ultimately we are interested in the number of cycles among the relations found. Alas, the dependence of the number of cycles on the number (and type) of relations is poorly understood (cf. Section 2). As noted, $\pi(z_{\mathbf{r}}) + \pi(z_{\mathbf{a}})$ relations always suffice, and in past experiments the number of relations collected was always somewhat lower. Here, the estimated number of relations is $0.49 \cdot (\pi(z_{\mathbf{r}}) + \pi(z_{\mathbf{a}}))$. Using $\ell_{\mathbf{a}}, \ell_{\mathbf{r}} > 2$, as in the aforementioned experiments, would further increase the relation yield. Note that there are $T(y_{\mathbf{r}}, y_{\mathbf{a}})/23.2$ $\mathfrak{f}\mathfrak{f}$ s, which seems very reasonable.

It is worth observing that while the most ‘fertile’ area of the sieving region is close to the origin, the relation yield of the sieving region is not yet ‘dried out’: for example, doubling S to $6\text{E}23$ increases the number of relations significantly, to $2.8\text{E}10$. The practical significance is that if someone builds a TWIRL device with hard-wired smoothness bounds and (for whatever reason) does not find enough relations using the above parameters, recovery may be possible simply by increasing S , i.e., by sieving for a longer time using the same hardware.

Candidates. For integer k , let $\mu(y, k) = k/\eta(y, k)$ denote the non- y -smooth cofactor of k . Sieving per se (i.e., the task handled by TWIRL) merely identifies the pairs (a, b) for which $\mu(y_{\mathbf{r}}, N_{\mathbf{r}}(a, b)) \leq z_{\mathbf{r}}^{\ell_{\mathbf{r}}}$ and $\mu(y_{\mathbf{a}}, N_{\mathbf{a}}(a, b)) \leq z_{\mathbf{a}}^{\ell_{\mathbf{a}}}$. For $\ell_{\mathbf{a}} = \ell_{\mathbf{r}} = 2$, not all such pairs form relations. Thus subsequent filtering is applied, and it should be verified that its cost is manageable. Also, in the ‘cascaded sieves’ variant employed by the revised TWIRL design, the algebraic-side sieve handles only the pairs (a, b) that passed the rational sieve, and it should be

verified that the latter are sufficiently infrequent (cf. [18, A.6]; this is crucial for achieving the high parallelism factor of 32768 inspected pairs per clock cycle). Thus, we estimate the number of candidates at the relevant points in the algorithm by writing down the appropriate probability, integrating it over the sieving region and multiplying the result by the correction factor $6/\pi^2$ (cf. Section 3).

The types of candidates are listed below; the results of the integrations are given in Table 10. In the following, let k_1, k_2 ($k_1 \geq k_2$) denote the two largest prime factors of $N_{\mathbf{r}}(a, b)$, and let κ_1, κ_2 ($\kappa_1 \geq \kappa_2$) denote the two largest prime factors of $N_{\mathbf{a}}(a, b)$.

Pass rational sieve (PRS): The pairs that pass the rational sieve are those that fulfill $\mu(y_{\mathbf{r}}, N_{\mathbf{r}}(a, b)) \leq z_{\mathbf{r}}^2$. Noting that $z_{\mathbf{r}}^2 < z_{\mathbf{a}}^3$, we get that the above is equivalent to the following: $(k_1, k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1 \leq z_{\mathbf{r}}^2 \wedge k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1, k_2 \wedge k_1 k_2 \leq z_{\mathbf{r}}^2)$. Accordingly, the probability that (a, b) fulfills this can be estimated by $\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}/2) + \bar{\sigma}_2(u_{\mathbf{r}}, v_{\mathbf{r}}/2, v_{\mathbf{r}}/2)$.

Pass both sieves (PBS): the probability that a pair (a, b) passes both sieves is obtained by multiplying the above by the analogous expression for the algebraic side: $(\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}/2) + \bar{\sigma}_2(u_{\mathbf{r}}, v_{\mathbf{r}}/2, v_{\mathbf{r}}/2)) \cdot (\rho(u_{\mathbf{a}}) + \sigma_1(u_{\mathbf{a}}, v_{\mathbf{a}}/2) + \bar{\sigma}_2(u_{\mathbf{a}}, v_{\mathbf{a}}/2, v_{\mathbf{a}}/2))$.

Pass primality testing (PPT): For pairs that passed both sieves, the smooth factors are divided out to obtain $\mu(y_{\mathbf{r}}, N_{\mathbf{r}}(a, b))$ and $\mu(y_{\mathbf{a}}, N_{\mathbf{a}}(a, b))$ (note that most prime factors smaller than $y_{\mathbf{r}}$ or $y_{\mathbf{a}}$ are reported by TWIRL). If $\mu(y_{\mathbf{r}}, N_{\mathbf{r}}(a, b))$ is prime and $> z_{\mathbf{r}}$, or $\mu(y_{\mathbf{r}}, N_{\mathbf{a}}(a, b))$ is prime and $> z_{\mathbf{a}}$, then the pair is discarded. A pair (a, b) reaches and survives this test iff $(k_1, k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1 \leq z_{\mathbf{r}} \wedge k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1, k_2 \wedge k_1 k_2 \leq z_{\mathbf{r}}^2)$ and analogously for the algebraic side. The probability that this holds is estimated by $(\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}) + \bar{\sigma}_2(u_{\mathbf{r}}, v_{\mathbf{r}}/2, v_{\mathbf{r}}/2)) \cdot (\rho(u_{\mathbf{a}}) + \sigma_1(u_{\mathbf{a}}, v_{\mathbf{a}}) + \bar{\sigma}_2(u_{\mathbf{a}}, v_{\mathbf{a}}/2, v_{\mathbf{a}}/2))$.

Rational cofactor factorizations (RCF): For pairs that survived primality testing, if the cofactor $\mu(y_{\mathbf{r}}, N_{\mathbf{r}}(a, b))$ is composite then it needs to be factored and tested for $z_{\mathbf{r}}$ -smoothness. The size of the cofactor to be factored is bounded by $z_{\mathbf{r}}^2$. This step is reached and the factorization is performed if $(y_{\mathbf{r}} < k_1, k_2 \wedge k_1 k_2 \leq z_{\mathbf{r}}^2)$ and $(\kappa_1, \kappa_2 < y_{\mathbf{a}}) \vee (y_{\mathbf{a}} < \kappa_1 \leq z_{\mathbf{a}} \wedge \kappa_2 < y_{\mathbf{a}}) \vee (y_{\mathbf{a}} < \kappa_1, \kappa_2 \wedge \kappa_1 \kappa_2 \leq z_{\mathbf{a}}^2)$. The probability that this holds is estimated by $\bar{\sigma}_2(u_{\mathbf{r}}, v_{\mathbf{r}}/2, v_{\mathbf{r}}/2) \cdot (\rho(u_{\mathbf{a}}) + \sigma_1(u_{\mathbf{a}}, v_{\mathbf{a}}) + \bar{\sigma}_2(u_{\mathbf{a}}, v_{\mathbf{a}}/2, v_{\mathbf{a}}/2))$.

Rational semi-smooth (RSS): A pair reaches the rational cofactor factorization step and passes (or skips) it if indeed $N_{\mathbf{r}}(a, b)$ is $(y_{\mathbf{r}}, z_{\mathbf{r}}, \ell_{\mathbf{r}})$ -smooth and (a, b) passed the algebraic sieve. For this to happen, the condition on the rational side is $(k_1, k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1 \leq z_{\mathbf{r}} \wedge k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1, k_2 \leq z_{\mathbf{r}})$, and the condition on the algebraic side is as in the previous step. Thus the probability is estimated by $(\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}) + \sigma_2(u_{\mathbf{r}}, v_{\mathbf{r}})) \cdot (\rho(u_{\mathbf{a}}) + \sigma_1(u_{\mathbf{a}}, v_{\mathbf{a}}) + \bar{\sigma}_2(u_{\mathbf{a}}, v_{\mathbf{a}}/2, v_{\mathbf{a}}/2))$.

Algebraic cofactor factorizations (ACF): For pairs that passed all of the above, if the cofactor $\mu(y_{\mathbf{a}}, N_{\mathbf{a}}(a, b))$ is composite then it needs to be factored and tested for $z_{\mathbf{a}}$ -smoothness. This step is reached and the factorization is performed iff $(y_{\mathbf{a}} < \kappa_1, \kappa_2 \wedge \kappa_1 \kappa_2 \leq z_{\mathbf{a}}^2)$ and also the rational-side condition of the previous step holds. The corresponding probability is estimated by $(\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}) + \sigma_2(u_{\mathbf{r}}, v_{\mathbf{r}})) \cdot \bar{\sigma}_2(u_{\mathbf{a}}, v_{\mathbf{a}}/2, v_{\mathbf{a}}/2)$.

Relations (Total): A pair that passes all of the above forms a relation; the probability of this occurring is estimated by $(\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}) + \sigma_2(u_{\mathbf{r}}, v_{\mathbf{r}})) \cdot (\rho(u_{\mathbf{a}}) + \sigma_1(u_{\mathbf{a}}, v_{\mathbf{a}}) + \sigma_2(u_{\mathbf{a}}, v_{\mathbf{a}}))$.

The above describes one plausible ordering of the filtering steps; other variations are possible (e.g., performing the algebraic cofactor factorization before the rational cofactor factorization, or even before the rational primality testing).

Cost of Cofactor Factorization. As indicated above, we expect to perform about $\#RCF + \#ACF = 7.7\text{E}10$ factorizations of integers whose size is at most $\max(z_{\mathbf{r}}, z_{\mathbf{a}})^2 = 3.6\text{E}23$. Such factorizations require under 30ms on average using a modern CPU. Thus, the cofactor factorization can be completed in 1 year (i.e., in parallel to the operation of the TWIRL device) using about 74 bare-bones PCs. This cost is negligible compared to the cost of TWIRL, and in large volumes custom hardware would reduce it further.

Optimality and Effect of Technological Progress. The revised TWIRL parameters were essentially determined by practical concerns. Most crucially, they employ the largest value of $y_{\mathbf{a}}$ for which the algebraic-side TWIRL device still fits on single silicon wafer. Theoretically, this $y_{\mathbf{a}}$ is suboptimal; it would be beneficial to increase it. Such increase will become possible when progress in chip manufacturing technology allows fitting larger circuits into a single wafer, either by increasing the wafer size or by decreasing the feature size. Thus, for the foreseeable future we may expect the cost of TWIRL to decrease more than linearly as a function of the relevant technological parameters, i.e., faster than naively implied by Moore's law.

For a concrete example, one may consider an implementation of TWIRL using 90nm process technology, which is expected to be widely deployed during 2004. Compared to the 130nm process technology considered in [18], we may assume a reduction in area by a factor of 2 and an increase in speed by a factor of 2, for a total cost reduction by a factor of 4 (cf. [8]). Table 11 presents two appropriate NFS parameter sets. The first set is about as plausible as the one in Table 10; the cost of such a TWIRL implementation is roughly $1.1\text{M US\$} \times \text{year}$ (predicted analogously to [18]) — considerably lower than $2.5\text{M US\$} \times \text{year}$ one may expect.

The second parameter set in Table 11 shows the effect of improved technology on yield, when keeping the cost constant at $10\text{M US\$} \times \text{year}$ (i.e., the same as in [18]). Here, the estimated number of relations is $1.95 \cdot (\pi(z_{\mathbf{r}}) + \pi(z_{\mathbf{a}}))$, which is nearly twice the trivially sufficient number. Also, there are $T(y_{\mathbf{r}}, y_{\mathbf{a}})/3.6$ *ff*s, which is much more than in any recent factoring experiment. Thus, we may conclude that using 90nm technology, a budget of $10\text{M US\$} \times \text{year}$ per factorization (in large quantities) leaves an ample safety margin — arguably,

Table 11. RSA-1024 parameter sets for TWIRL with 90nm process technology.
$$y_r = 1.2\text{E}10, y_a = 5.5\text{E}10, z_r = 8.0\text{E}11, z_a = 1.0\text{E}12, T(y_r, y_a) \approx 2.9\text{E}9, S = 8.0\text{E}22$$

$$d = 5, s = 1991935.4, B = 1.4\text{E}8$$

yield of (L_a, Lr) -partial relations									
(0, 0)	(0, 1)	(0, 2)	(1, 0)	(1, 1)	(1, 2)	(2, 0)	(2, 1)	(2, 2)	Total
2.2E8	9.8E8	1.8E9	9.2E8	4.0E9	7.5E9	1.4E9	6.1E9	1.1E10	3.4E10

#PRS	#PBS	#PPT	#RCF	#RSS	#ACF	avg($N_r(a, b)$)	avg($N_a(a, b)$)
6.3E19	1.1E13	9.8E10	7.2E10	5.9E10	4.5E10	2.7E63	1.1E102

$$y_r = 1.2\text{E}10, y_a = 5.5\text{E}10, z_r = 9.0\text{E}11, z_a = 1.2\text{E}12, T(y_r, y_a) \approx 2.9\text{E}9, S = 7.3\text{E}23$$

$$d = 5, s = 1991935.4, B = 4.3\text{E}8$$

yield of (L_a, Lr) -partial relations									
(0, 0)	(0, 1)	(0, 2)	(1, 0)	(1, 1)	(1, 2)	(2, 0)	(2, 1)	(2, 2)	Total
7.9E8	3.9E9	7.9E9	3.4E9	1.7E10	3.4E10	5.4E9	2.7E10	5.5E10	1.5E11

#PRS	#PBS	#PPT	#RCF	#RSS	#ACF	avg($N_r(a, b)$)	avg($N_a(a, b)$)
5.2E20	4.6E13	4.6E11	3.4E11	2.7E11	2.1E11	8.1E63	2.8E104

Table 12. RSA-768 parameters and estimates for [18].
$$y_r = 1.0\text{E}8, y_a = 1.0\text{E}9, z_r = 2.0\text{E}10, z_a = 3.0\text{E}10, T(y_r, y_a) \approx 5.7\text{E}7, S = 3.0\text{E}20$$

$$d = 5, s = 1905116.1, B = 8.9\text{E}6$$

yield of (L_a, Lr) -partial relations									
(0, 0)	(0, 1)	(0, 2)	(1, 0)	(1, 1)	(1, 2)	(2, 0)	(2, 1)	(2, 2)	Total
3.5E6	2.2E7	5.5E7	2.5E7	1.5E8	3.9E8	6.2E7	3.8E8	9.7E8	2.1E9

#PRS	#PBS	#PPT	#RCF	#RSS	#ACF	avg($N_r(a, b)$)	avg($N_a(a, b)$)
5.3E17	3.4E11	7.5E9	6.3E9	3.9E9	3.2E9	3.4E49	7.1E82

more than enough to account for estimation errors, relations that are lost due to approximations in the sieving process, and sub-optimal cycles-finding algorithms.

Parameter Settings for 768-bit Composites. For RSA-768, [18] uses the following polynomial, obtained by the same method as above:

$$\begin{aligned}
d = 5: \quad & m = 2980427354552256959621694668022969720925142335553136586170340190386865951921 \\
& 42458430585097389943648179813292845509402284357573098406890616147678906706078002760 \\
& 825484610584689826591183386558993533887364961255454143572139671622998845, \\
& s = 1905116.1, t = \exp(3.78), \\
& f(X) = 44572350495893220X^5 \\
& \quad + 1421806894351742986806319X^4 \\
& \quad - 1319092270736482290377229028413X^3 \\
& \quad - 4549121160536728229635596952173101064X^2 \\
& \quad + 6062531470679201843447146909871507448641523X \\
& \quad - 1814356642608474735992878928235210850251713945286, \\
& g(X) = 669580586761796376057918067X - 7730028528962337116069068686542066657037329.
\end{aligned}$$

The parameter choice and yield estimates using this polynomial are given in Table 12.

Index Calculus Attack for Hyperelliptic Curves of Small Genus

Nicolas Thériault

University of Toronto

Abstract. We present a variation of the index calculus attack by Gaudry which can be used to solve the discrete logarithm problem in the Jacobian of hyperelliptic curves. The new algorithm has a running time which is better than the original index calculus attack and the Rho method (and other square-root algorithms) for curves of genus ≥ 3 . We also describe another improvement for curves of genus ≥ 4 (slightly slower, but less dependent on memory space) initially mentioned by Harley and used in a number of papers, but never analyzed in details.

1 Introduction

Koblitz [10] first introduced the use of hyperelliptic curves for discrete log based public-key cryptography in 1989. For the first ten years, the best known generic attacks against these cryptosystems were the “square-root” algorithms (Shank’s Baby Step-Giant Step, Pollard’s ρ method, Pollard’s λ method). Pier- rick Gaudry’s index calculus attack for hyperelliptic curves [8] was the first exam- ple of a generic attack that could solve the discrete log problem on the Jacobian of an hyperelliptic curve of small genus over a finite field in a time smaller than the square-root of the group order (assuming the genus of the curve is greater than 4) (an attack for curves of high genus was introduced the year before in [1] by Adleman, DeMarrais and Huang).

In this paper, we analyse in detail a variation of the original index calcu- lus attack which was first introduced by Robert Harley and implemented for a number of papers, but never analyzed in detail. This algorithm works in time $O\left(g^5 q^{2-\frac{2}{g+1}+\epsilon}\right)$ and gives an improvement on previous attacks for curves of genus greater than 3. We also describe how the algorithm can be improved fur- ther by using the large prime method of the number field sieve. For this variation, we get a running time of $O\left(g^5 q^{2-\frac{4}{2g+1}+\epsilon}\right)$ and an improvement for all curves of genus greater than 2. Comparing the running times for curves of genus 3, 4 and 5, we get

g	3	4	5
square-root algorithms	$q^{3/2}$	q^2	$q^{5/2}$
original index calculus	q^2	q^2	q^2
reduced factor base	$q^{3/2}$	$q^{8/5}$	$q^{5/3}$
with large primes	$q^{10/7}$	$q^{14/9}$	$q^{18/11}$

This paper is divided as follows: The main ideas and concepts used in the index calculus attack are described in Sect. 2. We then present the two algorithms in Sects. 3 and 4. The running times of both algorithms are analyzed together in Sect. 5 and the memory space required to run the algorithms is discussed in Sect. 6.

2 The Index Calculus Attack

2.1 The Discrete Log Problem

Let C be an imaginary quadratic curve over \mathbb{F}_q , i.e. a smooth hyperelliptic curve of genus g over \mathbb{F}_q with a single point at infinity and whose finite part can be written in the form $y^2 + h(x)y = f(x)$ with $\deg(f) = 2g + 1$ and $\deg(h) \leq g$.

Note 1. Throughout this paper, we will use J_q for $Jac(C)(\mathbb{F}_q)$.

Definition 1. Given D_1, D_2 , two elements of J_q such that $D_2 \in \langle D_1 \rangle$, the hyperelliptic discrete log problem for the pair (D_1, D_2) on J_q consists in computing the smallest integer $\lambda \in \mathbb{N}$ such that $D_2 = \lambda D_1$.

In practice, we can assume that D_1 has large prime order in J_q (if not, we can bring the problem down to subgroups of $\langle D_1 \rangle$ of prime order, solving the corresponding discrete log problem on each subgroup independently).

2.2 Jacobian Arithmetic

Note 2. Throughout the paper, we will assume only basic arithmetic for multiplication. In practice, faster algorithms (Karatsuba, FFT) should be used, but they will reduce the overall running time by a factor of less than $g \log(q)$.

Points of $Jac(C)$ can be represented uniquely by reduced divisors, i.e. divisors of the form

$$\sum_{i=1}^k P_i - k\infty$$

where the P_i 's are points in $C(\overline{\mathbb{F}_q})$ with $P_i \neq -P_j$ for $i \neq j$ and with $k \leq g$ and ∞ is the unique point at infinity of C . From now onward, we identify $Jac(C)$ with the collection of reduced divisors.

We use the following result of Cantor [2]:

Proposition 1. For every reduced divisor $D = \sum_{i=1}^k P_i - k\infty$ (with $P_i = (x_i, y_i)$), there is a unique representation by a pair of polynomials $[a(x), b(x)]$, $a(x), b(x) \in \overline{\mathbb{F}_q}[x]$, with

$$a(x) = \prod_{i=1}^k (x - x_i)$$

and $b(x_i) = y_i$ satisfying $\deg(b) < \deg(a) \leq g$ and $b(x)^2 + h(x)b(x) - f(x)$ divisible by $a(x)$. The sum (as a reduced divisor) of two reduced divisors in J_q can be computed in $O(g^2(\log(q))^2)$ bit operations.

The reduced divisor $D = [a(x), b(x)]$ is associated to a point in J_q if and only if both $a(x)$ and $b(x)$ are in $\mathbb{F}_q[x]$.

2.3 Smooth Divisors

Let \mathcal{P} be the collection of \mathbb{F}_q -rational points of C , i.e. $\mathcal{P} = C(\mathbb{F}_q)$. For every $P \in C(\overline{\mathbb{F}_q})$, we let $D(P) = P - \infty$.

Definition 2. Let B be a subset of \mathcal{P} . A divisor D is said to be smooth relative to B if it is reduced and $D = \sum_{i=1}^k D(P_i)$ with all the P_i 's in B .

Definition 3. A subset B of \mathcal{P} used to define smoothness is called a factor base.

Definition 4. A divisor will be said to be potentially smooth if it is smooth relative to \mathcal{P} .

Definition 5. A point P in \mathcal{P} will be called a large prime relative to a factor base B if $P \notin B$.

Definition 6. A reduced divisor $D = \sum_{i=1}^k D(P_i)$ will be said to be almost-smooth if all but one of the P_i 's are in B and the remaining P_i is a large prime.

2.4 Random Walk

The index calculus algorithm relies in a large part on using a pseudo-random walk to search for smooth divisors. We set up a pseudo-random walk by specifying a hash function \mathcal{H} and a state function \mathcal{R} . A hash function \mathcal{H} is a function $\mathcal{H} : J_q \rightarrow \{1, 2, \dots, n\}$. A state function is a map $\mathcal{R} : J_q \times \{1, 2, \dots, n\} \rightarrow J_q$. Given an initial point $T_0 \in J_q$, our interest is in computing the sequence (the "random walk") (T_i) with $T_{i+1} = \mathcal{R}(T_i, \mathcal{H}(T_i))$.

To have an effective index calculus attack for the discrete log problem for a given pair $(D_1, D_2) \in J_q \times J_q$, the pair $(\mathcal{R}, \mathcal{H})$ should be chosen to satisfy certain statistical and computational constraints. The function \mathcal{R} should be chosen so that given $T_i = \alpha_i D_1 + \beta_i D_2$, it is easy to compute T_{i+1} as well as α_{i+1} and β_{i+1} such that $T_{i+1} = \alpha_{i+1} D_1 + \beta_{i+1} D_2$. A simple method is to set

$$\mathcal{R}(T, j) = T + T^{(j)}$$

where $T^{(j)} = \alpha^{(j)} D_1 + \beta^{(j)} D_2$ for some randomly chosen $\alpha^{(j)}$ and $\beta^{(j)}$.

At each step of the random walk, we compute T_{i+1} as well as α_{i+1} and β_{i+1} modulo the order of J_q . The values of T_i , α_i and β_i need to be recorded only if T_i is a smooth divisor (or an almost-smooth divisor in the second algorithm).

2.5 Index Calculus

From the sequence (T_i) of divisors obtained in the random walk, we extract a subsequence of smooth divisors (R_i) . Then each R_i can be written both as $R_i = \alpha_i D_1 + \beta_i D_2$ and $R_i = \sum_{j=1}^{k_i} D(P_j)$ with the P_j 's in the factor base and

$k_i \leq g$. The goal of the index calculus attack is to use the R_i 's to obtain an equation of the form $\alpha D_1 + \beta D_2 = 0$.

To do this, we order the elements of B as $P_1, P_2, \dots, P_{|B|}$. To each smooth divisor

$$T_i = \sum_{j=1}^{k_i} D(P_{i,j}) = \sum_{l=1}^{|B|} a_{i,l} D(P_l)$$

we can associate a vector

$$\mathbf{v}_i = (a_{i,1}, a_{i,2}, \dots, a_{i,|B|})$$

We then use the vectors \mathbf{v}_i to build the matrix $M = (a_{i,j})_{i,j}$ where each row, corresponding to a smooth divisor, has weight at most g . When the size of M is large enough (i.e. when M is overdetermined), we use linear algebra to find a nonzero vector in the kernel of M . Note that all operations are done modulo $|J_q|$. Once a nonzero solution of the system is found, we can write

$$\sum_{i=0}^m \gamma_i \mathbf{v}_i = \mathbf{0}$$

and (in terms of divisors)

$$\sum_{i=0}^m \gamma_i R_i = 0.$$

Substituting $R_i = \alpha_i D_1 + \beta_i D_2$, we get

$$\left(\sum_{i=0}^m \gamma_i \alpha_i \right) D_1 + \left(\sum_{i=0}^m \gamma_i \beta_i \right) D_2 = \alpha D_1 + \beta D_2 = 0,$$

from which we obtain the solution $D_2 = \lambda D_1$ ($\lambda = -\alpha/\beta$). The algorithm fails only if $\beta = 0$, in which case we must go through the algorithm again starting from the initialization of the random walk. This is very unlikely however (the algorithm fails with probability $|\langle D_1 \rangle|^{-1}$ if D_1 has prime order), hence we expect to have to go through the algorithm only once.

In practice, once a point $P_i \neq -P_i$ is included in the factor base, we take $-P_i$ as being in the factor base but replace $D(-P_i)$ by $-D(P_i)$ in the construction of the linear algebra system (since the divisor $D(P_i) + D(-P_i)$ reduces to 0). This makes it possible to reduce the number of smooth divisors we must find in the random walk by a factor of close to 2.

3 First Algorithm

3.1 Factor Base

In the original version of the index calculus attack for hyperelliptic curve, the factor base is $\mathcal{P} = C(\mathbb{F}_q)$. This gives a running time of

$$O(g^2 g! q (\log(q))^2) + O(g^3 q^2 (\log(q))^2)$$

where the first part is due to the search for smooth divisors, while the second part is the cost of solving the linear algebra system.

If q is large enough relative to g , i.e. if $q > (g-1)!$, then most of the cost of the index calculus attack comes from the linear algebra. The first approach to reduce the overall running time consists in reducing the size of the factor base, which reduces the time required to solve the linear algebra system on the one hand, but increases the search time on the other hand (since reducing the size of the factor base also reduces the number of smooth divisors). We do this until both parts of the running time are equal, i.e. up to the point where any further reduction of the factor base would make the search too costly.

Given that $q > (g-1)!$, the factor base can be chosen as a subset B of \mathcal{P} such that the running time becomes

$$O\left(g^5 q^{2-\frac{2}{g+1}+\epsilon}\right).$$

For the analysis, we assume that $q > (g-1)!$ and we set $|B| = q^r$, with $\frac{2}{3} < r < 1$ and compute the value of r which gives the best running time.

3.2 Algorithm

The first algorithm can be summarized as follows:

1. Search for the elements of the factor base
Compute the x and y coordinates of points in $C(\mathbb{F}_q)$ until $|B| = q^r$.
2. Initialization of the random walk
Choose the $\alpha^{(j)}$'s and $\beta^{(j)}$'s randomly and compute the $T^{(j)}$'s. Also choose α_0 and β_0 randomly and compute $T_0 = \alpha_0 D_1 + \beta_0 D_2$.
3. Search for smooth divisors (random walk)
The following steps are repeated until the linear system is large enough:
 - a) Search for potentially smooth divisors
Compute $T_{i+1} = [a(x), b(x)]$ and check if $a(x)$ splits over \mathbb{F}_q .
 - b) Factorization of the potentially smooth divisors
If $a(x)$ splits over \mathbb{F}_q , compute the points in $C(\mathbb{F}_q)$ corresponding to T_{i+1} . T_{i+1} is smooth if and only if all the points are in B .
 - c) Construction of the linear algebra system
Compute α_{i+1} and β_{i+1} . If T_{i+1} is smooth, record α_{i+1} , β_{i+1} and the factors of T_{i+1} .
4. Solution of the linear algebra system
Compute a nonzero vector in the kernel of the matrix obtained at step 3.
5. Final solution
Compute λ (if $\beta = 0$, return to step 2).

Note that in step 3, the factorization of $a(x)$ is done in two parts: we first check if $a(x)$ splits over \mathbb{F}_q by breaking down $a(x)$ into squarefree factors and checking that the factors divide $x^q - x$. If $a(x)$ splits in \mathbb{F}_q , we can then completely factor $a(x)$ using Cantor-Zassenhaus. The second part, which is probabilistic, is obviously skipped if $a(x)$ does not split over \mathbb{F}_q (in that case, the divisor is not potentially smooth and obviously cannot be smooth).

4 Second Algorithm

The new improvement to the index calculus mimics the use of large primes in the number field sieve. We again reduce the size of the factor base as much as possible to reduce the time required to solve the linear algebra system without making the search for smooth divisors too costly. This time however, we make use of the points in \mathcal{P} which are not part of the factor base.

If $q > (g-1)!/g$, we can play the almost-smooth divisors against each other to cancel the large primes to bring the running time down to

$$O\left(g^5 q^{2-\frac{4}{2g+1}+\epsilon}\right).$$

For the analysis, we once again assume that $q > (g-1)!/g$ and that the factor base has size $|B| = q^r$ with $\frac{2}{3} < r < 1$.

4.1 Large Primes

To make use of the almost-smooth divisors, we consider them in the order in which they appear during the search.

Definition 7. Let T_i be an almost-smooth divisor with the large prime P . T_i will be called an intersection if one of the previous T_j ($j < i$) has large prime $\pm P$.

If two almost-smooth divisors T_1, T_2 have large prime P , i.e. if they can be written in the form

$$T_1 = D(P) + \sum_{i=1}^{k_1-1} D(P_{1,i}) \quad \text{and} \quad T_2 = D(P) + \sum_{i=1}^{k_2-1} D(P_{2,i})$$

with $P_{1,i}, P_{2,i} \in B$, we consider

$$T_1 - T_2 = \sum_{i=1}^{k_1-1} D(P_{1,i}) - \sum_{i=1}^{k_2-1} D(P_{2,i})$$

and set $T' = T_1 - T_2$ (after doing all the extra cancellations that may be necessary if $P_{1,i} = P_{2,j}$ for some pair i, j).

If T_1, T_2 are almost-smooth divisors such that T_1 has large prime P and T_2 has large prime $-P$, i.e. if they can be written in the form

$$T_1 = D(P) + \sum_{i=1}^{k_1-1} D(P_{1,i}) \quad \text{and} \quad T_2 = D(-P) + \sum_{i=1}^{k_2-1} D(P_{2,i})$$

with $P_{1,i}, P_{2,i} \in B$, we consider

$$T_1 + T_2 = \sum_{i=1}^{k_1-1} D(P_{1,i}) + \sum_{i=1}^{k_2-1} D(P_{2,i})$$

and set $T' = T_1 + T_2$ (after doing all the extra cancellations that may be necessary if $P_{1,i} = -P_{2,j}$ for some pair i, j).

In both cases, T' factors over the factor base even though it may not be smooth (T' need not be reduced). For the linear algebra, the vector associated with T' will work in exactly the same way as if it was the difference of two smooth divisors with a common $P_i \in B$ and will have weight $< 2g$.

Proposition 2. *Each intersection is counted only once no matter how many times the large prime (or its negative) appeared before.*

Proof. Let P be a large prime. Suppose that $k > 1$ almost-smooth divisors with large prime P or $-P$ occurred during the random walk, say $T_{j_1}, T_{j_2}, \dots, T_{j_k}$, $k - 1$ of which are intersections. Using the same idea as described in Sect. 2.5, we associate the T_{j_i} 's to vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k$ with an extra coordinate for $D(P)$. In order to use these to add information to the linear algebra system, we must cancel out the coordinate associated to $D(P)$. If we use \mathbf{v}_1 to do the cancellation in the other \mathbf{v}_i 's, we obtain $k - 1$ vectors $\mathbf{v}'_2, \dots, \mathbf{v}'_k$ which are then used to construct M (after removing the coordinate associated to $D(P)$). Since

$$\text{span}\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\} = \text{span}\{\mathbf{v}_1, \mathbf{v}'_2, \dots, \mathbf{v}'_k\},$$

once T_{j_1} has been used to cancel the large prime in T_{j_i} , using another T_{j_i} to do the cancellation again does not produce any supplementary information for the linear algebra system. Q.E.D.

We therefore look for intersections of almost-smooth divisors and use these to obtain extra equations in our linear algebra system.

The advantage of this method is that the number of almost-smooth divisors is greater than the number of smooth divisors by a factor of $O(gq^{1-r})$ and the search should produce more intersections of almost-smooth divisors than smooth divisors.

For the analysis, we will assume that any point P_i in \mathcal{P} such that $P_i = -P_i$ is in the factor base and that a point is in the factor base if and only if its negative is also in the factor base. This has no effect on the running time, but it simplifies the analysis (in particular for Theorem 1).

4.2 Algorithm

The second algorithm can be summarized as follows (all steps, except 3c, work in the same way as in the first algorithm).

1. Search for the elements of the factor base
2. Initialization of the random walk

3. Search for smooth divisors (random walk)
 - a) Search for potentially smooth divisors
 - b) Factorization of the potentially smooth divisors
 - c) Cancellation of the Large Primes

If the divisor is almost-smooth, check whether or not it is an intersection.
If not, add it to the list of non-intersections. If it is an intersection, cancel its large prime and use the result as if it were a smooth divisor.
 - d) Construction of the linear algebra system
4. Solution of the linear algebra system
5. Final solution

5 Running-Time Analysis

5.1 Factor Base

In order to choose our factor base, we look at the x -coordinates of the points in $C(\mathbb{F}_q)$.

We go through the values of $x_i \in \mathbb{F}_q$ starting from 0 and following a chosen order on \mathbb{F}_q . We first evaluate $y^2 + h(X)y - f(X)$ at $X = x_i$ (this can be done in $O(g^2(\log(q))^2)$ bit operations). We then factor the quadratic polynomial in $\mathbb{F}_q[y]$ obtained which takes $O((\log(q))^2)$ bit operations. If the polynomial has roots $y_{i,1}, y_{i,2}$ in \mathbb{F}_q (y_i if we have a double root), we include $(x_i, y_{i,1})$ and $(x_i, y_{i,2})$ in B . We then go on to the next $x_i \in \mathbb{F}_q$ until $|B| = q^r$.

This method will require $O(q)$ tries for the possible x -coordinates, each taking $O(g^2(\log(q))^2)$ bit operations, for a total of

$$O(g^2 q (\log(q))^2)$$

bit operations to build the factor base.

5.2 Initialization

To initialize the Random walk, we need to precompute the divisors $T^{(i)}$ used in the state function $\mathcal{R} : J_q \times \{0, 1, \dots, n\} \rightarrow J_q$ as well as T_0 .

For each $T^{(i)}$ (and for T_0), we choose both $\alpha^{(i)}$ and $\beta^{(i)}$ randomly in $\{1, 2, \dots, (|J_q| - 1)\}$ and set $T^{(i)} = \alpha^{(i)}D_1 + \beta^{(i)}D_2$. We then need $O(g \log(q))$ Jacobian operations to compute each of the $T^{(i)}$'s, each Jacobian operation taking $O(g^2(\log(q))^2)$ bit operations. In practice, we can take $n = O(\log(|J_q|)) = O(g \log(q))$, which gives a total of

$$O(g^4 \log(q)^4)$$

bit operations to initialize the random walk.

5.3 Smooth Divisors

Proposition 3. *For $\frac{2}{3} < r < 1$, there are $\frac{q^{rg}}{g!} + O\left(\frac{q^2 q^{r(g-1)}}{g!}\right)$ smooth divisors in J_q .*

Proof. All smooth divisors relative to B can be written in the form $\sum_{i=1}^k D(P_i)$ with the P_i 's in B and $k \leq g$. To count to number of smooth divisors, we need to consider the number of distinct P_i 's in the representation of the divisors. The number of smooth divisors with g distinct P_i 's is:

$$\frac{1}{g!} \prod_{i=0}^{g-1} (q^r - i) = \frac{q^{rg}}{g!} - \frac{q^{r(g-1)}}{2(g-2)!} + O\left(q^{r(g-2)}\right).$$

The number of smooth divisors with $g-1$ distinct P_i 's, one of which is repeated is:

$$\frac{g-1}{(g-1)!} \prod_{i=0}^{g-2} (q^r - i) = \frac{q^{r(g-1)}}{(g-2)!} + O\left(q^{r(g-2)}\right).$$

The number of smooth divisors with $g-1$ distinct P_i 's, none of which are repeated is:

$$\frac{1}{(g-1)!} \prod_{i=0}^{g-2} (q^r - i) = \frac{q^{r(g-1)}}{(g-1)!} + O\left(q^{r(g-2)}\right)$$

Finally, the number of smooth divisors with less than $g-1$ distinct P_i 's is $O(q^{r(g-2)})$. This gives a total of

$$\frac{q^{rg}}{g!} + O\left(\frac{q^2 q^{r(g-1)}}{g!}\right)$$

smooth divisors relative to B . Q.E.D.

The proportion of smooth divisors in J_q is then

$$\frac{\frac{q^{rg}}{g!} + O\left(\frac{q^2 q^{r(g-1)}}{g!}\right)}{q^g + O\left(gq^{g-\frac{1}{2}}\right)} = \frac{q^{-(1-r)g}}{g!} + O\left(\frac{q^2 q^{-(1-r)g-r}}{g!}\right) + O\left(\frac{gq^{-(1-r)g-\frac{1}{2}}}{g!}\right),$$

so we expect to have to look at

$$O\left(g! q^{(1-r)g}\right)$$

divisors for each smooth divisor found in the search.

5.4 Potentially Smooth Divisors

Proposition 4. *For $\frac{2}{3} < r < 1$, there are $\frac{q^g}{g!} + O\left(\frac{q}{g!} q^{g-\frac{1}{2}}\right)$ potentially smooth divisors in J_q .*

Proof. All smooth divisors relative to \mathcal{P} can be written in the form $\sum_{i=1}^k D(P_i)$ with the P_i 's in \mathcal{P} and $k \leq g$. To count the number of smooth divisors, we need to consider the number of distinct P_i 's in the representation of the divisors. Since $|\mathcal{P}| = q + O(\sqrt{q})$ (from Hasse's bound), the number of potentially smooth divisors with g distinct P_i 's is:

$$\frac{1}{g!} \prod_{i=0}^{g-1} (|\mathcal{P}| - i) = \frac{q^g}{g!} + O\left(\frac{g}{g!} q^{g-\frac{1}{2}}\right).$$

The number of potentially smooth divisors with less than g distinct P_i 's is $O(q^{g-1})$, which gives a total of

$$\frac{q^g}{g!} + O\left(\frac{g}{g!} q^{g-\frac{1}{2}}\right)$$

potentially smooth divisors. Q.E.D.

The proportion of potentially smooth divisors in J_q is then

$$\frac{\frac{q^g}{g!} + O\left(\frac{g}{g!} q^{g-\frac{1}{2}}\right)}{q^g + O\left(g q^{g-\frac{1}{2}}\right)} = \frac{1}{g!} + O\left(\frac{g}{g! \sqrt{q}}\right)$$

and we expect to have a potentially smooth divisor for every $O(g!)$ divisors computed in the search.

5.5 Almost-Smooth Divisors

Proposition 5. *For $\frac{2}{3} < r < 1$, there are $\frac{q^{rg+1-r}}{(g-1)!} + O\left(\frac{q^{rg}}{(g-1)!}\right)$ almost-smooth divisors in J_q .*

Proof. Each almost-smooth divisor can be written in the form $D(P) + \sum_{i=1}^{k-1} D(P_i)$ with $P \in \mathcal{P} \setminus B$, the P_i 's in B and $k \leq g$, so each almost-smooth divisor can be associated to a large prime and at most $g-1$ P_i 's in B . Using an argument similar to the one in the proof of Proposition 3, we get

$$\frac{q^{r(g-1)}}{(g-1)!} + O\left(\frac{(g-1)^2 q^{r(g-2)}}{(g-1)!}\right)$$

possible distinct choices for the P_i 's in B . There are $|\mathcal{P}| - |B| = q - q^r + O(\sqrt{q})$ choices for the large prime, so we have

$$\frac{q^{rg+1-r}}{(g-1)!} - \frac{q^{rg}}{(g-1)!} + O\left(\frac{(g-1)q^{rg+1-2r}}{(g-2)!}\right) + O\left(\frac{q^{rg+\frac{1}{2}-r}}{(g-1)!}\right)$$

almost-smooth divisors relative to B . Since $\frac{2}{3} < r < 1$ and $q > g!$, we get

$$\frac{q^{rg+1-r}}{(g-1)!} + O\left(\frac{q^{rg}}{(g-1)!}\right).$$

Q.E.D.

The proportion of almost-smooth divisors in J_q is

$$\frac{\frac{q^{rg+1-r}}{(g-1)!} + O\left(\frac{q^{rg}}{(g-1)!}\right)}{q^g + O\left(gq^{g-\frac{1}{2}}\right)} = \frac{q^{-(1-r)(g-1)}}{(g-1)!} + O\left(\frac{q^{-(1-r)g}}{(g-1)!}\right) + O\left(g \frac{q^{-(1-r)(g-1)-\frac{1}{2}}}{(g-1)!}\right).$$

During the search, we can expect to look at

$$O\left((g-1)!q^{(1-r)(g-1)}\right)$$

divisors for each almost-smooth divisor found.

5.6 Intersections

We now consider the effect on the search of using almost-smooth divisors to get the equations required for the linear algebra more quickly.

In order to know how many equations can be obtained from the almost-smooth divisors, we need an estimate of the expected number of intersections out of a set of s almost-smooth divisors. For this, we consider only the large prime of each almost-smooth divisor.

Let $Q(n, s, i)$ be the probability of having i intersections out of a sample of size s drawn with replacement from a set of n elements and let $E_{n,s}$ be the expected number of intersections, i.e.

$$E_{n,s} = \sum_{i=0}^{s-1} iQ(n, s, i).$$

Theorem 1. *If $3 \leq s < n/2$, then $E_{n,s}$ is between $\frac{2s^2}{3n}$ and $\frac{s^2}{n}$.*

Proof. If we consider the probability of having i intersections after $s+1$ draws, we have

$$Q(n, s+1, i) = \frac{n-2(s-i)}{n}Q(n, s, i) + \frac{2(s-i+1)}{n}Q(n, s, i-1)$$

since if T_{s+1} contains the large prime P_{s+1} , then T_{s+1} is an intersection if and only if $\pm P_{s+1}$ appears in one of the $s-i$ or $s-i+1$ non-intersections in the first i almost-smooth divisors. Then

$$\begin{aligned} E_{n,s+1} &= \sum_{i=0}^s iQ(n, s+1, i) \\ &= \sum_{i=0}^s i \left(\frac{n-2(s-i)}{n}Q(n, s, i) + \frac{2(s-i+1)}{n}Q(n, s, i-1) \right) \\ &= \sum_{i=0}^{s-1} i \frac{n-2(s-i)}{n}Q(n, s, i) + \sum_{i=1}^s i \frac{2(s-i+1)}{n}Q(n, s, i-1) \end{aligned}$$

$$\begin{aligned}
&= \frac{n-2s}{n} \sum_{i=0}^{s-1} iQ(n, s, i) + \frac{2}{n} \sum_{i=0}^{s-1} i^2 Q(n, s, i) + \frac{2s}{n} \sum_{i=1}^s Q(n, s, i-1) \\
&\quad + \frac{2s-2}{n} \sum_{i=1}^s (i-1)Q(n, s, i-1) - \frac{2}{n} \sum_{i=1}^s (i-1)^2 Q(n, s, i-1) \\
&= \frac{n-2}{n} \sum_{i=0}^{s-1} iQ(n, s, i) + \frac{2s}{n} \sum_{i=0}^{s-1} Q(n, s, i) \\
&= \frac{n-2}{n} E_{n,s} + \frac{2s}{n}.
\end{aligned}$$

Solving for $E_{n,s}$ (using $E_{n,1} = 0$), we get

$$E_{n,s} = \frac{n}{2} \left(1 - \frac{2}{n}\right)^s + s - \frac{n}{2} = \frac{n}{2} \sum_{i=2}^s \binom{s}{i} \left(\frac{-2}{n}\right)^i$$

Since $\frac{2(s-i)}{in} < 1$, the terms in the sum are strictly decreasing in absolute values, hence

$$E_{n,s} < \frac{s(s-1)}{n} < \frac{s^2}{n}$$

and

$$E_{n,s} > \frac{s(s-1)}{n} - \frac{2s(s-1)(s-2)}{3n^2} > \frac{s^2}{n} - \frac{2s^3}{3n^2} = \frac{s^2}{n} \left(1 - \frac{2}{3} \frac{s}{n}\right) > \frac{2}{3} \frac{s^2}{n}.$$

Q.E.D.

5.7 Search (First Algorithm)

In order to insure the existence of a nonzero vector in the kernel of the linear algebra system in step 4, we need to find $O(|B|) = O(q^r)$ smooth divisors. Since we expect to look at $O(g!q^{g(1-r)})$ divisors for each smooth divisor found, the search will take an expected

$$O\left(g!q^{g(1-r)+r}\right)$$

random walk steps.

At each step of the random walk, we first have to compute T_i which requires $O(g^2(\log(q))^2)$ bit operations for the arithmetic in J_q . From the representation of T_i as $[a(x), b(x)]$, we can test whether or not T_i is potentially smooth by checking if $a(x)$ factors into linear factors over \mathbb{F}_q , which can be done in $O(g^2 \log(q)^2)$ bit operations. We must also compute α_i and β_i modulo $|J_q|$, which requires $O(g^2(\log(q))^2)$ bit operations. Since this must be done for all $O(g!q^{g(1-r)+r})$ divisors generated, this gives

$$O\left(g^2 g! q^{g(1-r)+r} \log(q)^2\right)$$

bit operations.

We now consider the cost of completely factoring the potentially smooth divisors. Since there are $O(g!)$ divisors for each potentially smooth divisor, we expect to find $O(q^{g(1-r)+r})$ potentially smooth divisors during the search. Since computing the points of \mathcal{P} in the representation of a divisor $[a(x), b(x)]$ requires to completely factor $a(x)$ over \mathbb{F}_q (to get the x -coordinates and multiplicities) and then evaluating $b(x)$ at the roots of $a(x)$ (to obtain the y -coordinates), which takes $O(g^2 \log(q)^2)$ bit operations (since $a(x)$ has degree $O(g)$), determining which potentially smooth divisors are really smooth and representing them in terms of the factor base takes

$$O\left(g^2 q^{g(1-r)+r} \log(q)^2\right)$$

bit operations.

The search is then expected to take

$$O\left(g^2 g! q^{g(1-r)+r} \log(q)^2\right)$$

bit operations for the first algorithm.

Note that it may be possible to reduce the number of divisors to consider for factorization by giving conditions on the coefficients of $a(x)$ for the divisor to be considered for smoothness. For example, if q is prime and the x -coordinates of the points in the factor base are between 0 and cq^r , then if the divisor is smooth, $a(x)$ must be of the form $x^k - a_{k-1}x^{k-1} + \dots$ with $0 < a_{k-1} < kcq^r$. Even though this reduces the cost of testing for potentially smooth divisors and complete factorization, the arithmetic in J_q is unaffected, and so the effect on the running time will be at most a constant factor. This method will not work for the second algorithm since there are no restrictions on the x -coordinate of the large prime.

5.8 Search (Second Algorithm)

If we let n be the number of large primes (i.e. $n = q - q^r + O(\sqrt{q})$) and ask that $E_{n,s} = O(q^r)$ (i.e. so that we expect the search to yield enough intersections to build the linear algebra system), then we need

$$s = O\left(q^{\frac{r+1}{2}}\right).$$

It will then take

$$O\left(s(g-1)!q^{(g-1)(1-r)}\right) = O\left((g-1)!q^{(g-1)(1-r)+\frac{r+1}{2}}\right)$$

steps of random walk to build the linear algebra system.

Note that we expect the search to also produce

$$\frac{O\left((g-1)!q^{g-rg+r-1+\frac{r+1}{2}}\right)}{O(g!q^{g-rg})} = O\left(\frac{1}{g}q^{r-\frac{1-r}{2}}\right)$$

smooth divisors, which are obviously used to get the linear algebra system but are not enough to have an important effect on the running time.

As in the first algorithm, computing $T_i = [a(x), b(x)]$, α_i and β_i and testing whether or not T_i is potentially smooth takes $O(g^2 \log(q)^2)$, for a total of

$$O\left(gg!q^{(g-1)(1-r)+\frac{r+1}{2}}(\log(q))^2\right)$$

bit operations over the whole search.

Since one in every $O(g!)$ divisors is potentially smooth, we expect to find $O\left(q^{(g-1)(1-r)+\frac{r+1}{2}}/g\right)$ potentially smooth divisors during the search. For each potentially smooth divisor, we compute the points in \mathcal{P} in its representation (which takes $O(g^2 \log(q)^2)$ bit operations) and check if it is smooth or almost-smooth. If the divisor is smooth, it is used to produce the linear algebra system; if it is almost-smooth we look at the previous almost smooth divisors to see if it is an intersection, which takes $O(\frac{1+r}{2} \log(q))$ bit operations (there are $O(q^{\frac{1+r}{2}})$ non-intersections and only the large prime is considered doing this search). If we have an intersection, we cancel the large prime and use the resulting divisor to increase the size of the linear system, otherwise we add the divisor to the list of non-intersections. This process is expected to take

$$O\left(gq^{(g-1)(1-r)+\frac{r+1}{2}}(\log(q))^2\right)$$

bit operations for all the potentially smooth divisors encountered during the search.

The search is then expected to take

$$O\left(gg!q^{(g-1)(1-r)+\frac{r+1}{2}}(\log(q))^2\right)$$

bit operations for the second algorithm.

5.9 Linear Algebra

As said before, we continue with the search until we have an overdetermined system. This gives us a matrix M of size $O(q^r) \times O(q^r)$, hence there exists a nonzero vector in the kernel of M . Since each row has weight $O(g)$ ($\leq g$ for the first algorithm and $< 2g$ for the second), the system is sparse with weight $O(gq^r)$.

Since M is sparse, we can use the algorithms by Lanczos [11] and Wiedemann [13]. We can then find a vector in the kernel of this matrix in $O(gq^{2r})$ operations modulo $|J_q|$. Since $|J_q| = q^g + O(gq^{g-1/2})$, finding a solution will take

$$O\left(g^3 q^{2r} (\log(q))^2\right)$$

bit operations.

5.10 Final Solution

From the vector in the kernel of M , we have

$$\sum_i \gamma_i \mathbf{v}_i = \mathbf{0}.$$

We obtain the final solution by computing

$$\alpha = \sum_i \gamma_i \alpha_i \quad \text{and} \quad \beta = \sum_i \gamma_i \beta_i$$

modulo $|J_q|$, where α_i, β_i come from the representation as $T_i = \alpha_i D_1 + \beta_i D_2$ of the i^{th} divisor used to build the linear algebra system. If $\beta \neq 0$, the final solution of the discrete log problem for the pair (D_1, D_2) is

$$\lambda \equiv -\frac{\alpha}{\beta} \pmod{|J_q|}.$$

Computing α and β requires $O(q^r)$ operations modulo $|J_q|$, each of these operations taking $O(g^2(\log(q))^2)$ bit operations. This gives a total of

$$O(g^2 q^r (\log(q))^2)$$

bit operations for the final step.

5.11 Optimization (First Algorithm)

Theorem 2. *The factor base can be chosen such that the running time of the first algorithm becomes*

$$O\left(g^5 q^{2 - \frac{2}{g+1} + \epsilon}\right).$$

Proof. From the previous sections, the steps of the first algorithm have the following running times:

1. $O(g^2 q (\log(q))^2)$
2. $O(g^4 (\log(q))^4)$
3. $O(g^2 g! q^{g-(g-1)r} (\log(q))^2)$
4. $O(g^3 q^{2r} (\log(q))^2)$
5. $O(g^2 q^r (\log(q))^2)$

Since the running times for parts 1, 2 and 5 are all much smaller than those for parts 3 and 4 when $\frac{2}{3} < r < 1$, the overall running time is:

$$O\left(g^2 g! q^{g-(g-1)r} (\log(q))^2\right) + O\left(g^3 q^{2r} (\log(q))^2\right).$$

In order to minimize this, we choose r such that both parts have the same asymptotic form, i.e. such that

$$(g-1)! q^{g-(g-1)r} = q^{2r}.$$

Solving for r , we get

$$r = \frac{g + \log_q((g-1)!)}{g+1},$$

and since r is indeed between $\frac{2}{3}$ and 1 for genus ≥ 3 , this gives a running time of

$$O\left(g^3((g-1)!)^{\frac{2}{g+1}} q^{\frac{2g}{g+1}} (\log(q))^2\right).$$

Finally, since $(g/4)^{g+1} < (g-1)! < g^{g+1}$ for $g \geq 3$, this is

$$O\left(g^5 q^{2 - \frac{2}{g+1} + \epsilon}\right).$$

Q.E.D.

5.12 Optimization (Second Algorithm)

Theorem 3. *The factor base can be chosen such that the running time of the second algorithm becomes*

$$O\left(g^5 q^{2 - \frac{4}{2g+1} + \epsilon}\right).$$

Proof. For the second algorithm, the steps have running times:

1. $O(g^2 q (\log(q))^2)$
2. $O(g^4 (\log(q))^4)$
3. $O(gg! q^{(g-1)(1-r) + \frac{r+1}{2}} (\log(q))^2)$
4. $O(g^3 q^{2r} (\log(q))^2)$
5. $O(g^2 q^r (\log(q))^2)$

Once again, steps 3 and 4 are more costly than the others, so the overall running time is:

$$O\left(gg! q^{(g-1)(1-r) + \frac{r+1}{2}} (\log(q))^2\right) + O\left(g^3 q^{2r} (\log(q))^2\right).$$

Forcing both parts to have the same asymptotic form requires

$$(g-1)! q^{(g-1)(1-r) + \frac{r+1}{2}} = g q^{2r},$$

which gives

$$r = \frac{g - \frac{1}{2} + \log_q((g-1)!/g)}{g + \frac{1}{2}},$$

and since r is indeed between $\frac{2}{3}$ and 1 for genus ≥ 3 , this gives a running time of

$$O\left(g^3((g-1)!/g)^{\frac{4}{2g+1}} q^{\frac{4g-2}{2g+1}} (\log(q))^2\right).$$

Finally, since $(g/4)^{g+\frac{1}{2}} < (g-1)!/g < g^{g+\frac{1}{2}}$ for $g \geq 3$, we get

$$O\left(g^5 q^{2 - \frac{4}{2g+1} + \epsilon}\right).$$

Q.E.D.

6 Memory Space

For both algorithms, storing the linear algebra system requires $O(gq^r \log(q))$ bits ($O(q^r)$ equations and for each equation, the factored divisor, α_i and β_i each take $O(g \log(q))$ bits). For the second algorithm, we must also store all the non-intersections almost-smooth divisors, which requires $O\left(gq^{\frac{1+r}{2}} \log(q)\right)$ bits (we expect to need $O(q^{\frac{1+r}{2}})$ almost-smooth divisors, each taking $O(g \log(q))$ bits to store the factorization, α_i and β_i), which will take more space than the linear algebra system.

Substituting r with the values found in the proofs of Theorems 2 and 3, we get $O\left(g^2 q^{\frac{g}{g+1} + \epsilon}\right)$ bits for the first algorithm and $O\left(g^2 q^{\frac{2g}{2g+1} + \epsilon}\right)$ bits for the second algorithm.

7 Conclusion

We have described two algorithms for the hyperelliptic curves discrete log problem which improve on previously published attacks. If we compare the running time of these two algorithms with those of the original index calculus and the various “square-root” algorithms (Baby Step-Giant Step, Pollard ρ , etc.), for small genus, we have:

g	3	4	5	6	7	8	9
square-root algorithms	$q^{3/2}$	q^2	$q^{5/2}$	q^3	$q^{7/2}$	q^4	$q^{9/2}$
original index calculus	q^2	q^2	q^2	q^2	q^2	q^2	q^2
reduced factor base	$q^{3/2}$	$q^{8/5}$	$q^{5/3}$	$q^{12/7}$	$q^{7/4}$	$q^{16/9}$	$q^{9/5}$
with large primes	$q^{10/7}$	$q^{14/9}$	$q^{18/11}$	$q^{22/13}$	$q^{26/15}$	$q^{30/17}$	$q^{34/19}$

Since the running times using large primes are slightly lower than previously published attacks, the large primes algorithm should be taken into account when designing any cryptosystem based on hyperelliptic curves of genus greater than 2. In particular, for curves of genus 3, the field of definition requires approximately 5% more bits of memory space for the curves to give the same level of security as they did when the best known attacks were the square root algorithms (obviously, the cost of the group operation will also increase in consequence). The 5% increase is due to the ratio $\log(q')/\log(q) \approx 21/20$ required for the index calculus attack for a genus 3 curve defined over $\mathbb{F}_{q'}$ to require the same expected running time as Pollard’s ρ algorithm for a genus 3 curve defined over the field \mathbb{F}_q .

Note that for genus 2 curves, Gaudry showed that the linear algebra system can be solved in linear time (see [7]). The best possible running time for the index calculus (using all the points over \mathbb{F}_q as the factor base) is then $O(q)$, which is the same as Pollard’s ρ method and the other square roots algorithms.

References

1. L. M. Adleman, J. DeMarrais, M.-D. Huang, A subexponential algorithm for discrete logarithms over hyperelliptic curves of large genus over $\text{GF}(q)$, *Theoret. Comput. Sci.*, **226**, no. 1-2, pp. 7-18, 1999.
2. D. G. Cantor, Computing in the Jacobian of an hyperelliptic curve, *Math. Comp.*, **48**(177), pp. 95-101, 1987.
3. A. Enge, Computing discrete logarithms in high-genus hyperelliptic jacobians in provably subexponential time, *Math. Comp.*, **71**, no. 238, pp. 729-742, 2002.
4. A. Enge, P. Gaudry, A general framework for subexponential discrete logarithm algorithms, *Acta Arith.*, **102**, no. 1, pp. 83-103, 2002.
5. A. Enge, A. Stein, Smooth ideals in hyperelliptic function fields, *Math. Comp.*, **71**, no. 239, pp. 1219-1230, 2002.
6. T. Garefalakis, D. Panario, The index calculus method using non-smooth polynomials, *Math. Comp.*, **70**, no 235, pp. 1253-1264, 2001.
7. P. Gaudry, *Algorithmique des courbes hyperelliptiques et applications à la cryptologie*, Thèse de doctorat de l'École polytechnique, 2000
8. P. Gaudry, An algorithm for solving the discrete log problem on hyperelliptic curves, *Advances in cryptology - EUROCRYPT 2000*, Springer-Verlag, LNCS 1807, pp. 19-34, 2000.
9. M. Girault, R. Cohen, M. Campana, A generalized birthday attack, *Advances in Cryptology - EUROCRYPT '88*, Springer-Verlag, LNCS 330, pp. 129-156, 1988.
10. N. Koblitz, Hyperelliptic cryptosystems, *J. of Cryptology*, **1**, pp. 139-150, 1989.
11. B. A. LaMacchia, A. M. Odlyzko, Solving large sparse linear systems over finite fields, *Advances in Cryptology - CRYPTO '90*, Springer-Verlag, LNCS 537, pp. 109-133, 1990.
12. V. Müller, A. Stein, C. Thiel, Computing discrete logarithms in real quadratic congruence function fields of large genus, *Math. Comp.*, **68**, no. 226, pp. 807-822, 1999.
13. D. H. Wiedemann, Solving sparse linear equations over finite fields, *IEEE Trans. Inform. Theory*, **IT-32**, no. 1, pp.54-62, 1986.

Parallelizing Explicit Formula for Arithmetic in the Jacobian of Hyperelliptic Curves

Pradeep Kumar Mishra and Palash Sarkar

Cryptology Research Group
Applied Statistics Unit
Indian Statistical Institute
203 B T Road, Kolkata-700108, India

Abstract. One of the recent thrust areas in research on hyperelliptic curve cryptography has been to obtain explicit formulae for performing arithmetic in the Jacobian of such curves. We continue this line of research by obtaining parallel versions of such formulae. Our first contribution is to develop a general methodology for obtaining parallel algorithm of any explicit formula. Any parallel algorithm obtained using our methodology is provably optimal in the number of multiplication rounds. We next apply this methodology to Lange's explicit formula for arithmetic in genus 2 hyperelliptic curve – both for the affine coordinate and inversion free arithmetic versions. Since encapsulated add-and-double algorithm is an important countermeasure against side channel attacks, we develop parallel algorithms for encapsulated add-and-double for both of Lange's versions of explicit formula. For the case of inversion free arithmetic, we present parallel algorithms using 4, 8 and 12 multipliers. All parallel algorithms described in this paper are optimal in the number of parallel rounds. One of the conclusions from our work is the fact that the parallel version of inversion free arithmetic is more efficient than the parallel version of arithmetic using affine coordinates.

Keywords: hyperelliptic curve cryptography, explicit formula, parallel algorithm, Jacobian, encapsulated add-and-double.

1 Introduction

Hyperelliptic curves present a rich source of abelian groups over which the discrete logarithm problem is believed to be difficult. Hence these groups can be used for implementation of various public key primitives.

The main operation in a hyperelliptic curve based primitive is scalar multiplication, which is the operation of computing mX , where m is an integer and X is a (reduced) divisor in the Jacobian of the curve. Any algorithm for scalar multiplication requires an efficient method of performing arithmetic in the Jacobian. This arithmetic essentially consists of two operations – addition and doubling of divisors.

The basic algorithm for performing arithmetic in the Jacobian of hyperelliptic curves is due to Cantor [1]. However, this algorithm is not sufficiently fast for

practical implementation. There has been extensive research on algorithms for efficient arithmetic. The main technique is to obtain so called “explicit formula” for performing addition and doubling. These explicit formulae are themselves composed of addition, multiplication, squaring and inversion operations over the underlying finite field. Moreover, these formulae are specific to a particular genus. Thus there are separate formulae for genus 2 and genus 3 curves. See Table 1 in Section 2 for more details.

In this paper, we consider the problem of parallel execution of explicit formula. An explicit formula can contain quite a few field multiplications and squarings. (In certain cases, this can even be 50 or more.) On the other hand, the number of inversions is usually at most one or two. An explicit formula usually also contains many field additions; however, the cost of a field addition is significantly less than the cost of a field multiplication or inversion. Hence the dominant operation in an explicit formula is field multiplication.

On inspection of different explicit formulae appearing in the literature there appear to be groups of multiplication operations that can be executed in parallel. Clearly the ability to perform multiplications in parallel will improve the speed of execution of the algorithm. This gives rise to the following question: *Given an explicit formula, what is the best parallel algorithm for computing the formula?*

Our first contribution is to develop a general methodology for obtaining parallel version of any explicit formula. The methodology guarantees that the obtained parallel version requires the minimum number of rounds. The methodology can be applied to any explicit formula appearing in the literature. (There could also be other possible applications.)

The most efficient explicit formula for performing arithmetic in the Jacobian of genus 2 curve is given in [11,12]. In [11], the affine coordinate representation of divisors is used and both addition and doubling involve a field inversion. On the other hand, in [12] the explicit formula is developed for inversion free arithmetic in the Jacobian.

Our second contribution is to apply our methodology to both [11] and [12]. For practical applications, it is necessary to consider resistance to side channel attacks. One important countermeasure is to perform a so-called encapsulated add-and-double algorithm (see [3,6,7] for details). We develop parallel versions of encapsulated add-and-double algorithm for both [11] and [12]. In many situations, the number of parallel multipliers available may be limited. To deal with such situations we present the encapsulated add-and-double algorithm using inversion free arithmetic using 4, 8 and 12 multipliers. For the affine version we have derived an algorithm using 8 multipliers. All of our algorithms are optimal parallel algorithms in the sense that no other parallel algorithm can perform the computation in lesser number of rounds.

Some of our results that we obtain are quite striking. For example, using 4 multipliers, we can complete the inversion free encapsulated add-and-double algorithm in 27 rounds and using 8 multipliers we can complete it in 14 rounds. The algorithm involves 108 multiplications. In the case of arithmetic using affine coordinates, the 8 multiplier algorithm will complete the computation in 11

Table 1. Complexity of Explicit Formulae.

Genus	Name/Proposed in	Characteristic	Cost(Add)	Cost(Double)
Genus 2	Cantor [19]	All	$3[i] + 70[m/s]$	$3[i] + 76[m/s]$
	Nagao [19]	Odd	$1[i] + 55[m/s]$	$1[i] + 55[m/s]$
	Harley [5]	Odd	$2[i] + 27[m/s]$	$2[i] + 30[m/s]$
	Matsuo et al [14]	Odd	$2[i] + 25[m/s]$	$2[i] + 27[m/s]$
	Miyamoto et al [17]	Odd	$1[i] + 26[m/s]$	$1[i] + 27[m/s]$
	Takahashi [23]	Odd	$1[i] + 25[m/s]$	$1[i] + 29[m/s]$
	Lange [11]	All	$1[i] + 22[m] + 3[s]$	$1[i] + 22[m] + 5[s]$
	Lange [12]	All	$40[m] + 6[s]$	$47[m] + 4[s]$
Genus 3	Nagao [19]	Odd	$2[i] + 154[m/s]$	$2[i] + 146[m/s]$
	Pelzl et al [20]	All	$1[i] + 70[m] + 6[s]$	$1[i] + 61[m] + 10[s]$
Genus 4	Pelzl et al [21]	All	$2[i] + 160[m] + 4[s]$	$2[i] + 193[m] + 16[s]$

rounds including an inversion round. Usually inversions are a few times costlier than multiplications, the actual figure being dependent upon exact implementation details. However, from our results it is clear that in general the parallel version of arithmetic using affine coordinates will be costlier than the parallel version of inversion free arithmetic.

2 Preliminaries of Hyperelliptic Curves

In this section, we give a brief overview of hyperelliptic curves. For details, readers can refer to [15]. Let K be a field and let \bar{K} be the algebraic closure of K . A *hyperelliptic curve* C of genus g over K is an equation of the form $C : v^2 + h(u)v = f(u)$ where $h(u)$ in $K[u]$ is a polynomial of degree at most g , $f(u)$ in $K[u]$ is a monic polynomial of degree $2g + 1$, and there are no singular points (u, v) in $\bar{K} \times \bar{K}$. Unlike elliptic curves, the points on the hyperelliptic curve do not form a group. The additive group on which the cryptographic primitives are implemented is the divisor class group. Each element of this group is a *reduced divisor*. The group elements have a nice canonical representation by means of two polynomials of small degree. The algorithms Koblitiz [8] proposed for divisor addition and doubling are known as Cantor's algorithms.

Spallek [22] made the first attempt to compute divisor addition by explicit formula for genus 2 curves over fields of odd characteristic. Harley [5] improved the running time of the algorithm in [22]. Gaudry and Harley [4] observed that one can derive different explicit formula for divisor operations depending upon the weight of the divisors. In 2000, Nagao [19] proposed two algorithms; one for polynomial division without any inversion and another for extended gcd computation of polynomials requiring only one inversion. Both these algorithms can be applied to Cantor's algorithm to improve efficiency. Lange [10] generalised Harley's approach to curves over fields of even characteristic. Takahashi [23] and Miyamoto, Doi, Matsuo, Chao and Tsujii [17] achieved further speed-up using Montgomery's trick to reduce the number of inversions to 1. For genus 2 curves, the fastest version of explicit formula for inversion free arithmetic is given in [12]

and the fastest version of explicit formula using affine coordinates is given in [11]. Lange has also proposed various co-ordinate systems and explicit formula for arithmetic of genus 2 curves over them. Interested readers can refer to [13]. For genus 3 curves Pelzl, Wollinger, Guajardo and Paar [20] have proposed explicit formula for performing arithmetic. For genus 4 curves, Pelzl, Wollinger and Paar have derived explicit formulae [21]. Curves of genus 5 and above are considered insecure for cryptographic use.

We summarise the complexity of various explicit formulae proposed in literature in Table 1. The cost generally correspond to the most general case. In the cost column, $[i]$, $[m]$, $[s]$ stand for the time taken by an inversion, a multiplication and a squaring in the underlying field respectively. The notation, $[m/s]$ stands for time of a square or multiplication. In the corresponding papers, multiplications and squarings have been treated to be of the same complexity.

3 General Methodology for Parallelizing Explicit Formula

An explicit formula for performing doubling (resp. addition) in the Jacobian of a hyperelliptic curve is an algorithm which takes one (resp. two) reduced divisor(s) as input and produces a reduced divisor as output. Also the parameters of the curve are available to the algorithm. The algorithm proceeds by a sequence of elementary operations, where each operation is either a multiplication or an addition or an inversion over the underlying field. In general the formulae involve one inversion. If there is one inversion, the inversion operation can be neglected and the parallel version can be prepared without it. Later, it can be plugged in as a separate round at an appropriate place. The same is true if the formula contains more than one inversions. Hence, we can assume that the formula is inversion-free. The cost of a field multiplication (or squaring) is significantly more than the cost of a field addition and hence the number of field multiplications is the dominant factor determining the cost of the algorithm. On inspection of the different explicit formulae available in the literature, it appears that there are groups of multiplication operations which can be performed in parallel. The ability to perform several multiplications in parallel can significantly improve the total computation time. So the key problem that we consider is the following: *Given an explicit formula, identify the groups of multiplication operations that can be performed in parallel.* In this section we develop a general methodology for solving this problem.

Let \mathcal{F} be an explicit formula. Then \mathcal{F} consists of multiplication and addition operations. Also several intermediate variables are involved. First we perform the following preprocessing on \mathcal{F} .

1. *Convert all multiplications to binary operation* : Operations which are expressed as a product of three or more variables are rewritten as a sequence of binary operations. For example, the operation $p_5 = p_1 p_2 p_3$ is rewritten as $p_4 = p_1 p_2$ and $p_5 = p_3 p_4$.

2. *Reduce multiplication depth* : Suppose we are required to perform the following sequence of operations: $p_3 = p_1^2 p_2$; $p_4 = p_3 p_2$. The straightforward way of converting to binary results in the following sequence of operations: $t_1 = p_1^2$; $p_3 = t_1 p_2$; $p_4 = p_3 p_2$. Note that the three operations *have* to be done sequentially one after another. On the other hand, suppose we perform the operations in the following manner: $\{t_1 = p_1^2$; $t_2 = p_2^2\}$; $\{p_3 = t_1 p_2$; $p_4 = t_1 t_2\}$. In this case, the operations within $\{\}$ can be performed in parallel and hence the computation can be completed in two parallel rounds. The total number of operations increases to 4, but the number of parallel rounds is less. We have performed such operation using inspection. We also note that it should be fruitful to consider algorithmic approach to this step.
3. *Eliminate reuse of variable names* : Consider the following sequence of operations:

$$q_1 = p_1 + p_2; q_2 = p_3; \dots; q_1 = p_4 + p_5; \dots$$

In this case, at different points of the algorithm, the intermediate variable q_1 is used to store the values of both $p_1 + p_2$ and $p_4 + p_5$. During the process of devising the parallel algorithm we rename the variable q_1 storing the value of $p_4 + p_5$ by a unique new name. In the parallel algorithm we can again suitably rename it to avoid the overhead cost of initialising a new variable.

4. *Labeling process* : We assign unique labels to the addition and multiplication operations and unique names to the intermediate variables.

Given a formula \mathcal{F} , we define a directed acyclic graph $G(\mathcal{F})$ in the following fashion.

- The nodes of $G(\mathcal{F})$ correspond to the arithmetic operations and variables of \mathcal{F} . Also there are nodes for the parameters of the input divisor(s) as well as for the parameters of the curve.
- The arcs are defined as follows: Suppose $\mathbf{id} : r = qp$ is a multiplication operation. The identifier \mathbf{id} is the label assigned to this operation. Then the following arcs are present in $G(\mathcal{F})$: (q, \mathbf{id}) , (p, \mathbf{id}) and (\mathbf{id}, r) . Similarly, the arcs for the addition operations are defined, with the only difference being the fact that the indegree of an addition node may be greater than two.

Proposition 1. *The following are true for the graph $G(\mathcal{F})$.*

1. *The indegree of variable nodes corresponding to the parameters of the input divisors and the parameters of the curve is zero.*
2. *The indegree of any node corresponding to an intermediate variable is one.*
3. *The outdegree of any node corresponding to an addition or multiplication operation is one.*

Note that the outdegree of nodes corresponding to variables can be greater than one. This happens when the variable is required as input to more than one arithmetic operation. Our aim is to identify the groups of multiplication operations that can be performed in parallel. For this purpose, we prepare another graph $G^*(\mathcal{F})$ from $G(\mathcal{F})$ in the following manner:

- The nodes of $G^*(\mathcal{F})$ are the nodes of $G(\mathcal{F})$ which correspond to multiplication operation.
- There is an arc $(\mathbf{id}_1, \mathbf{id}_2)$ from node \mathbf{id}_1 to node \mathbf{id}_2 in $G^*(\mathcal{F})$ only if there is a path from \mathbf{id}_1 to \mathbf{id}_2 in $G(\mathcal{F})$ which does not pass through another multiplication node.

The graph $G^*(\mathcal{F})$ captures the ordering relation between the multiplication operations of \mathcal{F} . Thus, if there is an arc $(\mathbf{id}_1, \mathbf{id}_2)$ in $G^*(\mathcal{F})$, then the operation \mathbf{id}_1 must be done before the operation \mathbf{id}_2 . We now define a sequence of subgraphs of $G^*(\mathcal{F})$ and a sequence of subsets of nodes of $G^*(\mathcal{F})$ in the following manner.

- $G_1(\mathcal{F}) = G^*(\mathcal{F})$ and M_1 is the set of nodes of G_1 whose indegree is zero.
- For $i \geq 2$, G_i is the graph obtained from G_{i-1} by deleting the set M_{i-1} from G_{i-1} and M_i is the set of nodes of G_i whose indegree is zero.

Let r be the least positive integer such that G_{r+1} is the empty graph, i.e., on removing M_r from G_r , the resulting graph becomes empty.

Proposition 2. *The following statements hold for the graph $G^*(\mathcal{F})$.*

1. *The sequence M_1, \dots, M_r forms a partition of the nodes of $G^*(\mathcal{F})$.*
2. *All the multiplications in any M_i can be performed in parallel.*
3. *There is a path in $G^*(\mathcal{F})$ from some vertex in M_1 to some vertex in M_r . Consequently, at least r parallel multiplication rounds are required to perform the computation of \mathcal{F} .*

It is easy to obtain the sets M_i 's from the graph $G^*(\mathcal{F})$ by a modification of the standard topological sort algorithm [2]. The sets M_i ($1 \leq i \leq r$) represent only the multiplication operations of \mathcal{F} . To obtain a complete parallel algorithm, we have to organize the addition operations and take care of the intermediate variables. There may be some addition operations at the beginning of the formula. Since additions are to be performed sequentially, we can ignore these additions while deriving the parallelised formula, treating the sums they produce as inputs. Later, they can be plugged in at the beginning of the formula.

For $1 \leq i \leq r-1$, let A_i be the set of addition nodes which lie on a path from some node in M_i to some node in M_{i+1} . Further, let A_r be the set of addition nodes which lie on a path originating from some node in M_r . There may be more than one addition operation in a path from a node in M_i to a node in M_{i+1} . These additions have to be performed in a sequential manner. (Note that we are assuming that \mathcal{F} starts with a set of multiplication operations and ends with a set of addition operations. It is easy to generalize to a more general form.)

Each multiplication and addition operation produces a value which is stored in an intermediate variable. We now describe the method of obtaining the set of intermediate variables required at each stage of computation. Let I_1, \dots, I_{2r} and O_1, \dots, O_{2r} be two sequences of subsets of nodes of $G(\mathcal{F})$, where each I_i and O_j contain nodes of $G(\mathcal{F})$ corresponding to variables. The parameters of the curve and the input divisor(s) are not included in any of the I_i and O_j 's. These are assumed to be additionally present throughout the algorithm. For $1 \leq i \leq r$, these sets are defined as follows:

1. I_{2i-1} contains intermediate variables which are the inputs to the multiplication nodes in M_i .
2. I_{2i} contains intermediate variables which are the inputs to the addition nodes in A_i .
3. O_{2i-1} contains intermediate variables which are the outputs of the multiplication nodes in M_i .
4. O_{2i} contains intermediate variables which are the outputs of the addition nodes in A_i .

For $1 \leq j \leq 2r$, define

$$V_j = (\cup_{i=1}^j O_i) \cap (\cup_{i=j+1}^{2r} I_i). \quad (1)$$

If a variable x is in V_j , then it has been produced by some previous operation and will be required in some subsequent operation. We define the parallel version $\text{par}(\mathcal{F})$ of \mathcal{F} as a sequence of rounds

$$\text{par}(\mathcal{F}) = (\mathcal{R}_1, \dots, \mathcal{R}_r). \quad (2)$$

where $\mathcal{R}_i = (M_i, V_{2i-1}, A_i, V_{2i})$. In round i , the multiplications in M_i can be performed in parallel; the sets V_{2i-1} and V_{2i} are the sets of intermediate variables and A_i is the set of addition operations. Note that the addition operations are not meant to be performed in parallel. Indeed, in certain cases the addition operations in A_i have to be performed in a sequential manner. We define several parameters of $\text{par}(\mathcal{F})$.

Definition 1. Let $\text{par}(\mathcal{F}) = (\mathcal{R}_1, \dots, \mathcal{R}_r)$, be the r -round parallel version of the explicit formula \mathcal{F} . Then

1. The total number of multiplications (including squarings) occuring in $\text{par}(\mathcal{F})$ will be denoted by **TM**.
2. The multiplication width (**MW**) of $\text{par}(\mathcal{F})$ is defined to be $\text{MW} = \max_{1 \leq i \leq r} |M_i|$.
3. The buffer width (**BW**) of $\text{par}(\mathcal{F})$ is defined to be $\text{BW} = \max_{1 \leq i \leq 2r} |V_i|$.
4. A path from a node in M_1 to a node in M_r is called a critical path in $\text{par}(\mathcal{F})$.
5. The value r is the critical path length (**CPL**) of $\text{par}(\mathcal{F})$.

The parameter **MW** denotes the maximum number of multipliers that can operate in parallel. Using **MW** parallel multipliers \mathcal{F} can be computed in r parallel rounds. The buffer width **BW** denotes the maximum number of variables that are required to be stored at any stage in the parallel algorithm.

3.1 Decreasing the Multiplication Width

The method described above yeilds a parallel algorithm $\text{par}(\mathcal{F})$ for a given explicit formula \mathcal{F} . It also fixes the number of computational rounds r required to execute the algorithm using **MW** number of proessors. By definition, **MW** is the maximum number of multiplications taking place in a round. However, it may

happen that in many rounds the actual number of multiplications is less than MW. If we use MW multipliers, then some of the multipliers will be idle in such rounds. The most ideal scenario is $MW \approx \lceil TM/r \rceil$. However, such an ideal situation may not come about automatically. We next describe a method for making the distribution of the number of multiplication operations more uniform among various rounds.

We first prepare a *requirement table*. It is a table containing data about the intermediate variables created in the algorithm. For every variable it contains the name of the variables used in the expressions computing it, the latest round in which one of such variables is created and the earliest round in which the variable itself is used. For example, suppose an intermediate variable $v_x = v_y * v_z$ is computed in the j -th round. Of v_y and v_z , let v_z be the one which is computed later and in the i -th round. Let v_x be used earliest in the k -th round. Then in the requirement table we have an entry for v_x consisting of v_y, v_z, i, k . If both of v_x and v_y are input values then we may take $i = 0$. Note that we have $i < j < k$.

Now suppose, there are more than $\lceil TM/r \rceil$ multiplications in the j -th round. Further suppose that for some j_1 ($i + 1 \leq j_1 \leq k - 1$), the number of multiplications in the j_1^{th} round is less than $\lceil TM/r \rceil$. Then we transfer the multiplication producing v_x to the j_1^{th} round and hence reduce the multiplication width of the j -th round. This change of position of the multiplication operation does not affect the correctness of the algorithm.

This procedure is applied as many times as possible to rounds which contain more than $\lceil TM/r \rceil$ multiplications. As a result we obtain a parallel algorithm with a more uniform distribution of number of multiplication operations over the rounds and consequently reduces the value of MW.

3.2 Managing Buffer Width

The parameter BW provides the value of the maximum number of intermediate variables that is required to be stored at any point in the algorithm. This is an important parameter for applications where the amount of memory is limited. We justify that obtaining parallel version of an explicit formula does not substantially change the buffer width. Our argument is as follows.

First note that the total number of multiplications in the parallel version is roughly the same as the total number of multiplications in the original explicit formula. The only place where the number of multiplications increases is in the preprocessing step of reducing the multiplication depth. Moreover, the increase is only a few multiplications. The total number of addition operations remain the same in both sequential and parallel versions. Since the total numbers of multiplications and additions are roughly the same, the total number of intermediate variables also remains roughly the same.

Suppose that after round k in the execution of the parallel version, i intermediate variables have to be stored. Now consider a sequential execution of the explicit formula. Clearly, in the sequential execution, all operations upto round k has to be executed before any operation of round greater than k can be executed. The i intermediate variables that are required to be stored after round k

are required as inputs to operations in round greater than k . Hence these intermediate variables are also required to be stored in the sequential execution of the explicit formula.

4 Application to Lange's Explicit Formulae

In [11] and [12], Lange presented explicit formulae for addition and doubling in the Jacobian of genus 2 hyperelliptic curves. In fact, there are many special cases involved in these explicit formulae and our methodology can be applied to all the cases. But to be brief, we restrict our attention to the most general and frequent case only. The formulae in [11] uses an inversion each for addition and doubling while the formulae in [12] does not require any inversion.

We apply the methodology described in Section 3 separately to the formulae in [11] and [12]. In the case of addition, the inputs are two divisors D_1 and D_2 and in the case of doubling the input is only one divisor D_1 . We use the following conventions.

- We assume that the curve parameters $h_2, h_1, h_0, f_4, f_3, f_2, f_1, f_0$ are available to the algorithm.
- We do not distinguish between squaring and multiplication.
- The labels for the arithmetic operations in the explicit formula for addition start with **A** and the labels for the arithmetic operations in the explicit formula for doubling start with **D**. The second letter of the label (**M** or **A**) denotes (m)ultiplication or (a)ddition over the underlying field. Thus **AM23** denotes the 23rd multiplication in the explicit formula for addition.
- The intermediate variables for the explicit formula for addition are of the form p_i and the intermediate variables for the explicit formula for doubling are of the form q_j .
- In [11,12], multiplications by curve constants are presented. However, during the total multiplication count, some of these operations are ignored, since for most practical applications the related curve constants will be 0 or 1. In this section, we include the multiplication by the curve parameters. In Section 5, we consider the situation where these are 0 or 1.
- The set of intermediate variables (V_i 's) required at any stage is called the buffer state.

4.1 Inversion Free Arithmetic

In this section, we consider the result of application of the method of Section 3 to the inversion free formula for addition and doubling given in [12]. The details are presented in the Appendix. The details of addition formula is presented in Section A.1 and the details of the doubling formula is presented in Section A.2. We present a summary of the parameters of the parallel versions in Table 2.

Based on Table 2 and Proposition 2(3), we obtain the following result.

Theorem 1. *Any parallel algorithm for executing either the explicit formula for addition or the explicit formula for doubling presented in [12] will require at least 8 parallel multiplication rounds. Consequently, the parallel algorithms presented in Sections A.1 and A.2 are optimal algorithms.*

Table 2. Parameters for parallel versions of explicit formula in [12].

	MW	BW	CPL	TM
Add	8	20	8	59
Double	11	15	8	65

4.2 Arithmetic Using Affine Coordinates

The most efficient explicit formula for arithmetic using affine coordinates has been presented in [11]. Here we consider the result of applying the methodology of Section 3 to this formula. Again due to lack of space we present the details full version of the paper. The parallel version of the addition formula is presented therein.

A summary of the results is presented in Table 3.

Table 3. Parameters for parallel versions of explicit formula in [11].

	MW	BW	CPL	TM
Add	6	12	7*	29*
Double	5	13	8*	34*

* Including one inversion

We have the following result about the parallel versions of the explicit formula in [11].

Theorem 2. *Any parallel algorithm for executing the explicit formula for addition (resp. doubling) presented in [11] will require at least 7 (resp. 8) parallel multiplication rounds. Consequently, the parallel algorithms presented in [16] are optimal algorithms.*

5 Encapsulated Addition and Doubling Algorithm

In this section, we address several issues required for actual implementation.

- The algorithms of Section A include multiplications by the parameters of the curve. However, we can assume that $h_2 \in \{0, 1\}$. If $h_2 \neq 0$, then by substituting $y = h_2^5 y'$ and $x = h_2^2 x'$ and dividing the resulting equation by h_2^{10} , we can make $h_2 = 1$. Also, if the underlying field is not of characteristic 5, we can assume that $f_4 = 0$. Otherwise, we can make it so by substituting $x' = (x - f_4/5)$. In the algorithms presented below, we assume that $h_2 \in \{0, 1\}$ and $f_4 = 0$ and hence the corresponding multiplications are ignored. These decreases the total number of multiplications and hence also the number of parallel rounds. In most applications h_1, h_0 also are in $\{0, 1\}$. Hence efficiency in such situations can go up further. Thus all the operations in Section A of Appendix do not occur in the algorithms in this section.

- The usual add-and-double scalar multiplication algorithm is susceptible to side channel attacks. One of the main countermeasures is to perform both addition and doubling at each stage of scalar multiplication (see [3]). We call such an algorithm an encapsulated add-and-double algorithm. The parallel algorithms we present in this section are encapsulated add-and-double algorithms. All of them take as input two divisors D_1 and D_2 and produce as output $D_1 + D_2$ and $2D_1$.

5.1 Inversion Free Arithmetic

In this section, we consider parallel version of encapsulated add-and-double formula. We obtain the algorithms from the individual algorithms presented in Section A.1 and A.2.

First we note that the total number of multiplication operations for encapsulated add-and-double under the above mentioned conditions is 108. Since the value of MW for addition is 8 and for doubling is 11 and both have CPL = 8, a total of 19 parallel finite field multipliers can complete encapsulated addition and doubling in 8 parallel rounds. However, 19 parallel finite field multipliers may be too costly. Hence we describe algorithms with 4, 8 and 12 parallel multipliers. (Note that an algorithm with two multipliers is easy to obtain – we assign one multiplier to perform addition and the other to perform doubling.)

Suppose the number of multipliers is m and the total number of operations is TM. Then at least $\lceil (TM/m) \rceil$ parallel rounds are necessary. Any algorithm which performs the computation in these many rounds will be called a *best* algorithm. Our parallel algorithms with 4 and 8 multipliers are best algorithms. Further, our algorithm with 12 multipliers is optimal in the sense that no other parallel algorithm with 12 multipliers can complete the computation in less rounds.

The actual algorithms for performing inversion free arithmetic with 4 processors is presented in Table 5. Such tables for 8 and 12 processors are presented in the full version of the paper. This table only lists the multiplication and addition of field elements. Interested readers can access the full version of the paper at [16]. The labels in the table refer to the labels of operations in the algorithms in Section A.1 and A.2. We present a summary of the results in Table 2.

Table 4. Summary of algorithms with varying number of processors for inversion free arithmetic of [12].

No of Multipliers	2	4	8	12
Number of rounds	54	27	14	10

5.2 Affine Coordinates

An eight multiplier parallel version of explicit formula for encapsulated add-and-double is presented in the full version of the paper. In this case the total number of multiplications is 65. The eight multiplier algorithm requires 11 parallel rounds

Table 5. Computation chart using four parallel multipliers for inversion free arithmetic of [12].

Rnd	Operation
1	AM01, AM02, AM03, AM04
2	AM05, AM06, AM07, AM08
	AA01, AA02, AA03, AA04
3	DM01, DM02, DM04, DM08
	DA01, DA02, DA03, DA04
4	DM09, AM09, AM10, AM11
	DA05, DA06, DA07, AA07, AA08, AA09
5	AM12, AM13, AM14, AM16
	AA05, AA06
6	DM12, DM13, DM14, DM15
	DA08
7	DM16, DM17, DM18, DM19
	DA09, DA10
8	DM20, DM22, AM17, AM18
	AA10, DA11, DA11, DA12, DA13
9	AM19, AM20, AM21, AM22
	AA12, AA13, AA14, AA15
10	DM23, DM24, DM25, DM26
	DA14, DA15, DA16, DA17, DA18, DA19
11	DM27, DM29, AM23, AM24
12	AM25, AM26, AM27, AM28
13	AM29, AM30, DM30, DM31
	AA16, AA17
14	DM32, DM33, DM34, DM35
	DA20, DA21
15	AM31, AM32, AM33, AM34
	AA18, AA19
16	AM35, AM37, AM38, DM36
17	DM37, DM38, DM39, DM41
18	DM43, AM39, AM40, AM41
19	AM42, AM43, AM44, AM46
	AA20, AA21, AA22, AA23, AA24, AA25
20	DM44, DM45, DM46, DM47
21	DM48, DM49, DM50, AM47
	DA22, DA23, DA24, DA25
22	AM48, AM49, AM50, AM51
23	AM52, AM53, DM51, DM52
	AA26, AA27
24	DM53, DM54, DM55, DM56
25	DM57, AM54, AM55, AM56
	DA26, DA27, DA28
26	AM57, DM58, DM59, DM60
	AA28, AA29, AA30, AA31
27	DM62, DM63, DM65, DM66
	DA29, DA30, DA31, DA32, DA33, DA34

including an inversion round. On the other hand, the eight multiplier algorithm for inversion free arithmetic requires only 14 multiplication rounds. Thus, in general the parallel version of inversion free arithmetic will be more efficient than the parallel version of arithmetic obtained from affine coordinates.

6 Conclusion

In this work, we have developed a general methodology for deriving parallel versions of any explicit formula for computation of divisor addition and doubling. We have followed the methods to derive the parallel version of the explicit formula given in [12] and [11]. We have considered encapsulated add-and-double algorithms to prevent side channel attacks. Moreover, we have described parallel algorithms with different number of processors.

It has been shown that for the inversion free arithmetic of [12] and with 4, 8 and 12 field multipliers an encapsulated add-and-double can be carried out in 27, 14 and 10 parallel rounds respectively. All these algorithms are optimal in the number of parallel rounds. In the case of arithmetic using affine coordinates [11], an eight multiplier algorithm can perform encapsulated add-and-double using 11 rounds including an inversion round. Since an inversion is usually several times costlier than a multiplication, in general the parallel version of inversion free arithmetic will be more efficient than the parallel version of arithmetic using affine coordinates.

We have applied our general methodology to explicit formula for genus 2 curves. The same methodology can also be applied to the explicit formula for genus 3 curves and to other explicit formulae appearing in the literature. Performing these tasks will be future research problems.

References

1. D. G. Cantor. Computing in the Jacobian of a Hyperelliptic curve. In *Mathematics of Computation*, volume 48, pages 95-101, 1987.
2. T. H. Cormen, C. E. Leiserson and R. L. Rivest. *Introduction to Algorithms*, MIT Press, Cambridge, 1997.
3. J.-S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. *Proceedings of CHES 1999*, pp 292-302, 1999.
4. P. Gaudry and R. Harley Counting Points on Hyperelliptic Curves over Finite Fields. In *ANTS IV*, volume 1838 of LNCS; pp 297-312, Berlin, 2000, Springer-Verlag.
5. R. Harley. Fast Arithmetic on Genus 2 Curves. *Available at* <http://cristal.inria.fr/~harley/hyper>, 2000.
6. T. Izu and T. Takagi. A Fast Parallel Elliptic Curve Multiplication Resistant against Side-Channel Attacks Technical Report CORR 2002-03, University of Waterloo, 2002. Available at <http://www.cacr.math.uwaterloo.ca>.
7. T. Izu, B. Möller and T. Takagi. Improved Elliptic Curve Multiplication Methods Resistant Against Side Channel Attacks. *Proceedings of Indocrypt 2002*, LNCS 2551, pp 296-313.

8. N. Koblitz. Hyperelliptic Cryptosystems. In *Journal of Cryptology*, 1: pages 139–150, 1989.
9. A. J. Menezes, P. C. van Oorschot and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
10. T. Lange. Efficient Arithmetic on Hyperelliptic Curves. PhD thesis, Universität Gesamthochschule Essen, 2001.
11. T. Lange. Efficient Arithmetic on Genus 2 Curves over Finite Fields via Explicit Formulae. Cryptology ePrint Archive, Report 2002/121, 2002. <http://eprint.iacr.org/>.
12. T. Lange. Inversion-free Arithmetic on Genus 2 Hyperelliptic Curves. Cryptology ePrint Archive, Report 2002/147, 2002. <http://eprint.iacr.org/>.
13. T. Lange. Weighted Co-ordinates on Genus 2 Hyperelliptic Curves. Cryptology ePrint Archive, Report 2002/153, 2002. <http://eprint.iacr.org/>.
14. K. Matsuo, J. Chao and S. Tsujii. Fast Genus Two Hyperelliptic Curve Cryptosystems. In *ISEC2001, IEICE*, 2001.
15. A. Menezes, Y. Wu, R. Zuccherato. An Elementary Introduction to Hyperelliptic Curves. Technical Report CORR 96-19, University of Waterloo(1996), Canada. Available at <http://www.cacr.math.uwaterloo.ca>.
16. P. K. Mishra and P. Sarkar Parallelizing Explicit Formula in the Jacobian of Hyperelliptic Curves (Full Version) Available at the Technical Report Section (Number 16) of [www://isical.ac.in/~crg](http://www.isical.ac.in/~crg). Also available at IACR ePrint Archive, <http://eprint.iacr.org/>.
17. Y. Miyamoto, H. Doi, K. Matsuo, J. Chao and S. Tsujii. A fast addition algorithm for genus 2 hyperelliptic curves. In *Proc of SCIS2002, IEICE, Japan*, pp 497–502, 2002, in Japanese.
18. P. Montgomery. Speeding the Pollard and Elliptic Curve Methods for Factorisation. In *Math. Comp.*, vol 48, pp 243–264, 1987.
19. K. Nagao. Improving Group Law Algorithms for Jacobians of Hyperelliptic Curves. *ANTS IV, LNCS 1838*, Berlin 2000, Springer-Verlag.
20. J. Pelzl, T. Wollinger, J. Guajardo and C. Paar. Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves. Cryptology ePrint Archive, Report 2003/026, 2003. <http://eprint.iacr.org/>.
21. J. Pelzl, T. Wollinger and C. Paar. Low Cost Security: Explicit Formulae for Genus 4 Hyperelliptic Curves. Cryptology ePrint Archive, Report 2003/097, 2003. <http://eprint.iacr.org/>.
22. A. M. Spallek. Kurven vom Geschlecht 2 und ihre Anwendung in Public-Key-Kryptosystemen. PhD Thesis, Universität Gesamthochschule, Essen, 1994.
23. M. Takahashi. Improving Harley Algorithms for Jacobians of Genus 2 Hyperelliptic Curves. In *Proc of SCIS 2002, IEICE, Japan*, 2002, in Japanese.

A Details of Parallel Versions of Explicit Formula

The organisation of this section is as follows.

- Parallel version of the explicit formula for addition using inversion free arithmetic of [12] is presented in Section A.1.
- Parallel version of the explicit formula for doubling using inversion free arithmetic of [12] is presented in Section A.2.

Similar parallelised versions of addition and doubling algorithms for affine co-ordinates given in [11] have been derived using the methods presented in this paper and are available in the full version of the paper. Interested readers can find them at [16].

A.1 Addition Using Inversion Free Arithmetic

Algorithm

Input: Divisors $D_1 = [U_{11}, U_{10}, V_{11}, V_{10}, Z_1]$ and $D_2 = [U_{21}, U_{20}, V_{21}, V_{20}, Z_2]$.

Output: Divisor $D_1 + D_2 = [U'_1, U'_0, V'_1, V'_0, Z']$

Initial buffer: $U_{11}, U_{10}, V_{11}, V_{10}, Z_1, U_{21}, U_{20}, V_{21}, V_{20}, Z_2$.

Round 1

AM01. $Z = Z_1 Z_2$; **AM02.** $\tilde{U}_{21} = Z_1 U_{21}$; **AM03.** $\tilde{U}_{20} = Z_1 U_{20}$;

AM04. $\tilde{V}_{21} = Z_1 V_{21}$; **AM05.** $\tilde{V}_{20} = Z_1 V_{20}$; **AM06.** $p_1 = U_{11} Z_2$;

AM07. $p_2 = U_{10} Z_2$; **AM08.** $p_3 = V_{11} Z_2$.

Buffer: $Z, \tilde{U}_{21}, \tilde{U}_{20}, \tilde{V}_{21}, \tilde{V}_{20}, p_1, p_2, p_3$.

AA01. $p_4 = p_1 - \tilde{U}_{21}$; **AA02.** $p_5 = \tilde{U}_{20} - p_2$;

AA03. $p_6 = p_3 - \tilde{V}_{21}$; **AA04.** $p_7 = Z_1 + U_{11}$.

Buffer: $Z, \tilde{U}_{21}, \tilde{U}_{20}, \tilde{V}_{21}, \tilde{V}_{20}, p_3, p_4, p_5, p_6, p_{17}, p_7, Z$.

Round 2

AM09. $p_8 = U_{11} p_4$; **AM10.** $p_9 = Z_1 p_5$; **AM11.** $p_{10} = Z_1 p_4$;

AM12. $p_{11} = p_4^2$; **AM13.** $p_{12} = p_4 p_6$; **AM14.** $p_{13} = h_1 Z$;

AM15. $p_{14} = f_4 Z$; **AM16.** $p_{15} = V_{10} Z_2$

Buffer: $Z, \tilde{U}_{21}, \tilde{U}_{20}, \tilde{V}_{21}, \tilde{V}_{20}, p_{15}, p_3, p_4, p_5, p_{17}, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}$.

AA05. $p_{16} = p_{15} - \tilde{V}_{20}$;

AA06. $p_{17} = p_{16} + p_6$;

AA07. $p_{18} = p_8 + p_9$;

AA08. $p_{19} = p_{18} + p_{10}$;

AA09. $p_{20} = p_4 + \tilde{U}_{21}$;

Buffer: $Z, \tilde{U}_{21}, \tilde{U}_{20}, \tilde{V}_{21}, \tilde{V}_{20}, p_{15}, p_3, p_4, p_{17}, p_7, p_{12}, p_{13}, p_{14}, p_{18}, p_{19}, p_{20}$

Round 3

AM17. $p_{21} = p_5 p_{18}$; **AM18.** $p_{22} = p_{11} U_{10}$; **AM19.** $p_{23} = p_{19} p_{17}$;

AM20. $p_{24} = p_{18} p_{16}$; **AM21.** $p_{25} = p_{12} p_7$; **AM22.** $p_{26} = p_{12} U_{10}$;

Buffer: $Z, \tilde{U}_{21}, \tilde{U}_{20}, \tilde{V}_{21}, \tilde{V}_{20}, p_{15}, p_3, p_4, p_{13}, p_{14}, p_{20}, p_{21}, p_{22}, p_{23}, p_{24}, p_{25}, p_{26}$

AA10. $r = p_{21} + p_{22}$; **AA11.** $s_1 = p_{23} - p_{24} - p_{25}$;

AA12. $s_0 = p_{24} - p_{26}$;

AA13. $p_{27} = \tilde{U}_{21} + \tilde{U}_{20}$;

AA14. $p_{28} = p_{13} + 2\tilde{V}_{21}$;

AA15. $p_{29} = p_4 + 2\tilde{U}_{21} - p_{14}$;

Buffer: $Z, \tilde{U}_{21}, \tilde{U}_{20}, \tilde{V}_{21}, \tilde{V}_{20}, r, s_1, s_0, p_{15}, p_3, p_4, p_{20}, p_{27}, p_{28}, p_{29}$

Round 4

AM23. $R = Zr$; **AM24.** $s_0 = s_0 Z$; **AM25.** $s_3 = s_1 Z$;

AM26. $S = s_0 s_1$; **AM26.** $p_{30} = s_1 p_4$; **AM27.** $p_{31} = r p_{29}$;

AM28. $p_{32} = s_1 p_{28}$ **AM29.** $t = s_1 p_{20}$

Buffer: $\tilde{U}_{21}, \tilde{U}_{20}, \tilde{V}_{21}, \tilde{V}_{20}, r, s_1, s_0, R, s_3, S, t, p_{15}, p_3, p_4, p_{27}, p_{30}, p_{31}, p_{32}, p_{27}$

AA16. $p_{33} = s_0 - t$, **AA17.** $p_{34} = t - 2s_0$

Buffer: $\tilde{U}_{21}, \tilde{U}_{20}, \tilde{V}_{21}, \tilde{V}_{20}, r, s_1, s_0, R, s_3, S, p_{15}, p_3, p_4, p_{27}, p_{30}, p_{31}, p_{32}, p_{33}, p_{34}$

Round 5

AM30. $S_3 = s_3^2$; **AM31.** $\tilde{R} = R s_3$; **AM32.** $\tilde{S} = s_3 s_1$;

$$\mathbf{AM33.} \quad \tilde{S} = s_0 s_1; \quad \mathbf{AM34.} \quad l_0 = S\tilde{U}_{20}; \quad \mathbf{AM35.} \quad p_{35} = h_2 p_{33};$$

$$\mathbf{AM36.} \quad p_{36} = s_0^2; \quad \mathbf{AM37.} \quad p_{37} = R^2;$$

$$\mathbf{Buffer:} \quad \tilde{U}_{21}, \tilde{V}_{21}, \tilde{V}_{20}, l_2, l_0, S_3, \tilde{R}, \tilde{S}, \tilde{S}, S, p_{15}, p_3, p_{27}, p_{30}, p_{31}, p_{32}, p_{34}, p_{35}, p_{36}, p_{37}$$

$$\mathbf{AA18.} \quad p_{38} = \tilde{S} + S; \quad \mathbf{AA19.} \quad p_{39} = p_{35} + p_{32};$$

$$\mathbf{Buffer:} \quad \tilde{U}_{21}, \tilde{V}_{21}, \tilde{V}_{20}, l_2, l_0, S_3, \tilde{R}, \tilde{S}, \tilde{S}, p_{15}, p_3, p_{27}, p_{30}, p_{31}, p_{34}, p_{36}, p_{38}, p_{39}$$

Round 6

$$\mathbf{AM38.} \quad \tilde{R} = \tilde{R}\tilde{S}; \quad \mathbf{AM39.} \quad l_2 = \tilde{S}\tilde{U}_{21}; \quad \mathbf{AM40.} \quad p_{40} = p_{38}p_{27};$$

$$\mathbf{AM41.} \quad p_{41} = p_{30}p_{34}; \quad \mathbf{AM42.} \quad p_{42} = p_3\tilde{S}; \quad \mathbf{AM43.} \quad p_{43} = R p_{39};$$

$$\mathbf{AM44.} \quad p_{44} = h_2\tilde{R}; \quad \mathbf{AM45.} \quad p_{45} = p_{15}\tilde{R};$$

$$\mathbf{Buffer:} \quad \tilde{V}_{21}, \tilde{V}_{20}, l_2, l_0, S_3, \tilde{R}, \tilde{S}, \tilde{R}, p_{31}, p_{36}, p_{37}, p_{40}, p_{41}, p_{42}, p_{43}, p_{44}$$

$$\mathbf{AA20.} \quad l_1 = p_{40} - l_2 - l_0; \quad \mathbf{AA21.} \quad l_2 = l_2 + \tilde{S};$$

$$\mathbf{AA22.} \quad U'_0 = p_{36} + p_{41} + p_{42} + p_{43} + p_{31};$$

$$\mathbf{AA23.} \quad U'_1 = 2\tilde{S} - p_{45} + p_{44} - p_{37};$$

$$\mathbf{AA24.} \quad l_2 = l_2 - U'_1; \quad \mathbf{AA25.} \quad p_{46} = U'_0 - l_1;$$

$$\mathbf{Buffer:} \quad U'_0, U'_1, \tilde{V}_{21}, \tilde{V}_{20}, l_2, l_0, S_3, \tilde{R}, \tilde{S}, \tilde{R}, p_{46}$$

Round 7

$$\mathbf{AM46.} \quad p_{47} = U'_0 l_2; \quad \mathbf{AM47.} \quad p_{48} = S_3 l_0; \quad \mathbf{AM48.} \quad p_{49} = U'_1 l_2;$$

$$\mathbf{AM49.} \quad p_{50} = S_3 p_{46}; \quad \mathbf{AM50.} \quad Z' = \tilde{R}S_3; \quad \mathbf{AM51.} \quad U'_0 = \tilde{R}U'_0;$$

$$\mathbf{AM52.} \quad U'_1 = \tilde{R}U'_1;$$

$$\mathbf{Buffer state:} \quad U'_0, U'_1, \tilde{V}_{21}, \tilde{V}_{20}, \tilde{R}, p_{47}, p_{48}, p_{49}, p_{50}, Z'$$

$$\mathbf{AA26.} \quad p_{51} = p_{47} - p_{48}; \quad \mathbf{AA27.} \quad p_{52} = p_{49} + p_{50};$$

$$\mathbf{Buffer:} \quad U'_0, U'_1, \tilde{V}_{21}, \tilde{V}_{20}, \tilde{R}, p_{51}, p_{52}, Z'$$

Round 8

$$\mathbf{AM53.} \quad p_{53} = \tilde{R}\tilde{V}_{20}; \quad \mathbf{AM54.} \quad p_{54} = \tilde{R}\tilde{V}_{21}; \quad \mathbf{AM55.} \quad p_{55} = h_0 Z';$$

$$\mathbf{AM56.} \quad p_{56} = h_1 Z'; \quad \mathbf{AM57.} \quad p_{57} = h_2 U'_0; \quad \mathbf{AM58.} \quad p_{58} = h_2 U'_1;$$

$$\mathbf{Buffer state:} \quad U'_0, U'_1, p_{51}, p_{52}, p_{53}, p_{54}, p_{55}, p_{55}, p_{56}, p_{57}, p_{58}, Z'$$

$$\mathbf{AA28.} \quad p_{59} = p_{51} - p_{53} - p_{55}; \quad \mathbf{AA29.} \quad p_{60} = p_{52} - p_{54} - p_{56};$$

$$\mathbf{AA30.} \quad V'_0 = p_{57} + p_{59}; \quad \mathbf{AA31.} \quad V'_1 = p_{58} + p_{60};$$

$$\mathbf{Buffer state:} \quad U'_0, U'_1, V'_0, V'_1, Z'$$

A.2 Doubling Using Inversion Free Arithmetic

Algorithm

Input: Divisors $D_1 = [U_{11}, U_{10}, V_{11}, V_{10}, Z_1]$.

Output: Divisor $2D_1 = [U_1'', U_0'', V_1'', V_0'', Z_1'']$.

Initial Buffer: $U_{11}, U_{10}, V_{11}, V_{10}, Z_1$.

Round 1

DM01. $q_0 = Z_1^2$; **DM02.** $q_1 = h_1 Z_1$; **DM03.** $q_2 = h_2 U_{11}$;

DM04. $q_3 = h_0 Z_1$; **DM05.** $q_4 = h_2 U_{10}$; **DM06.** $q_5 = f_4 U_{11}$;

DM07. $q_6 = h_2 V_{11}$; **DM08.** $q_7 = f_2 Z_1$; **DM09.** $q_8 = V_{11} h_1$;

DM10. $q_9 = V_{10} h_2$; **DM11.** $q_{10} = f_4 U_{10}$;

Buffer: $q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}$

DA01. $\tilde{V}_1 = q_1 + 2V_{11} - q_2$; **DA02.** $\tilde{V}_0 = q_3 + 2V_{10} - q_4$;

DA03. $q_{11} = 2U_{10}$; **DA04.** $inv_1 = -\tilde{V}_1$; **DA05.** $q_{12} = q_7 - q_8 - q_9 - 2q_{10}$;

DA06. $q_{13} = 2q_{11} + q_{10} + q_6$; **DA07.** $q_{14} = q_{11} + 2q_7 + q_6$;

Buffer: $inv_1, \tilde{V}_1, \tilde{V}_0, q_0, q_{14}, q_{11} q_{12}, q_{13}$

Round 2

DM12. $q_{15} = V_{11}^2$; **DM13.** $q_{16} = U_{11}^2$; **DM14.** $q_{17} = \tilde{V}_0 Z_1$;

DM15. $q_{18} = U_{11} \tilde{V}_1$; **DM16.** $q_{19} = \tilde{V}_1^2$; **DM17.** $q_{20} = f_3 q_0$;

DM18. $q_{21} = q_{12} Z_1$; **DM19.** $q_{22} = q_{13} Z_1$; **DM20.** $q_{23} = q_{14} Z_1$;

DM21. $q_{24} = h_2 U_{11}$; **DM22.** $q_{25} = h_1 Z_1$;

Buffer: $inv_1, \tilde{V}_1, \tilde{V}_0, q_0, q_{15}, q_{16}, q_{17}, q_{18}, q_{19}, q_{20}, q_{21}, q_{22}, q_{23}, q_{24}, q_{25}$

DA08. $q_{26} = q_{17} q_{18}$; **DA09.** $q_{27} = q_{20} + q_{16}$; **DA10.** $q_{28} = q_{22} - q_{27}$;

DA11. $k_1 = 2q_{16} + q_{27} - q_{23}$; **DA12.** $q_{29} = q_{21} - q_{15}$;

DA13. $q_{30} = 2V_{10} - q_{24} + q_{25}$;

Buffer: $inv_1, \tilde{V}_0, k_1, q_0, q_{19}, q_{26}, q_{27}, q_{28}, q_{29}, q_{30}$

Round 3

DM23. $q_{31} = \tilde{V}_0 q_{26}$; **DM24.** $q_{32} = q_{19} U_{10}$; **DM25.** $q_{33} = U_{11} q_{28}$;

DM26. $q_{34} = Z_1 q_{29}$; **DM27.** $q_{35} = k_1 inv_1$; **DM28.** $q_{36} = f_4 Z_1$;

DM29. $q_{37} = Z_1 U_{10}$;

Buffer: $inv_1, k_1, q_0, q_{26}, q_{31}, q_{32}, q_{37}, q_{30}, q_{33}, q_{34}, q_{35}, q_{36}$

DA14. $r = q_{31} + q_{32}$; **DA15.** $k_0 = q_{33} + q_{34}$; **DA16.** $q_{38} = k_0 + k_1$;

DA17. $q_{39} = inv_1 + q_{26}$; **DA18.** $q_{40} = 1 + U_{11}$;

DA19. $q_{41} = 2U_{11} - q_{36}$;

Buffer: $q_0, r, k_0, q_{26}, q_{37}, q_{30}, q_{35}, q_{36}, q_{38}, q_{39}, q_{40}, q_{41}$

Round 4

DM30. $R = q_0 r$; **DM31.** $q_{42} = q_{38} q_{39}$; **DM32.** $q_{43} = q_{35} q_{40}$;

DM33. $q_{44} = q_{35} q_{37}$; **DM34.** $q_{45} = k_0 q_{26}$; **DM35.** $q_{46} = r q_{41}$;

Buffer: $R, q_{30}, q_{45}, q_{36}, q_{42}, q_{43}, q_{44}, q_{46}$

DA20. $s_3 = q_{42} - q_{45} - q_{43}$;

DA21. $s_0 = q_{45} - q_{44}$;

Buffer: $R, s_0, s_3, q_{30}, q_{46}$

Round 5

DM36. $q_{47} = R^2$; DM37. $q_{48} = s_0 s_3$; DM38. $s_1 = s_3 Z_1$;
 DM39. $S_0 = s_0^2$; DM40. $t = h_2 s_0$; DM41. $q_{49} = q_{30} s_3$;
 DM42. $q_{50} = h_2 R$; DM43. $q_{51} = Z_1 q_{46}$;
 Buffer: $S_0, t, s_1, q_{47}, q_{48}, q_{49}, q_{51}, q_{50}$

Addition phase

No addition required at this step.

Buffer: Same as above.

Round 6

DM44. $\tilde{R} = R s_1$; DM45. $S_1 = s_1^2$; DM46. $q_{52} = s_1 s_3$;
 DM47. $S = q_{48} Z_1$; DM48. $l_0 = U_{10} q_{48}$; DM49. $q_{53} = R q_{49}$;
 DM50. $q_{54} = q_{50} s_1$;
 Buffer: $\tilde{R}, S_1, S, S_0, t, l_0, q_{47}, q_{48}, q_{52}, q_{53}, q_{51}, q_{54}$
 DA22. $q_{55} = U_{11} + U_{10}$; DA23. $q_{56} = q_{48} + q_{52}$;
 DA24. $U_0'' = S_0 + q_{53} + t + q_{51}$;
 DA25. $U_1'' = 2S + q_{54} - q_{47}$;
 Buffer: $U_0'', U_1'', l_0, S_1, \tilde{R}, q_{55}, q_{52}, q_{56}$

Round 7

DM51. $\tilde{\tilde{R}} = \tilde{R} q_{52}$; DM52. $q_{57} = q_{56} q_{55}$; DM53. $q_{58} = S_1 l_0$;
 DM54. $Z'' = S_1 \tilde{R}$; DM55. $q_{59} = \tilde{R} U_1''$ DM56. $q_{60} = \tilde{R} U_0''$;
 DM57. $l_2 = U_{11} s_1$;
 Buffer: $U_0'', U_1'', Z'', \tilde{\tilde{R}}, S_1, l_0, l_1, l_2, q_{57}, q_{58}, q_{59}, q_{60}$
 DA26. $l_1 = q_{57} - l_2 - l_0$;
 DA27. $l_2 = l_2 + S - U_1''$; DA28. $q_{61} = U_0'' - l_1$;
 Buffer: $U_0'', U_1'', Z'', \tilde{\tilde{R}}, S_1, l_2, q_{58}, q_{59}, q_{60}, q_{61}$

Round 8

DM58. $q_{62} = U_0'' l_2$; DM59. $q_{63} = U_1'' l_2$; DM60. $q_{64} = S_1 q_{61}$;
 DM61. $q_{65} = h_2 q_{60}$; DM62. $q_{66} = \tilde{\tilde{R}} V_{10}$; DM63. $q_{67} = h_0 Z''$;
 DM64. $q_{68} = h_2 q_{59}$; DM65. $q_{69} = \tilde{\tilde{R}} V_{11}$; DM66. $q_{70} = h_1 Z''$;
 Buffer: $Z'', q_{58}, q_{59}, q_{60}, q_{62}, q_{63}, q_{64}, q_{65}, q_{66}, q_{67}, q_{68}, q_{69}, q_{70}$
 DA29. $q_{71} = q_{62} + q_{58}$; DA30. $q_{72} = q_{63} + q_{64}$;
 DA31. $U_0'' = q_{60}$; DA32. $U_1'' = q_{59}$;
 DA33. $V_0'' = q_{71} + q_{65} - q_{66} - q_{67}$;
 DA34. $V_1'' = q_{72} + q_{68} - q_{69} - q_{70}$;
 Buffer: $U_0'', U_1'', Z'', V_0'', V_1''$

Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$

Iwan Duursma¹ and Hyang-Sook Lee^{2,*}

¹ Department of Mathematics, University of Illinois at Urbana-Champaign
Urbana IL 61801, USA
duursma@math.uiuc.edu

² Department of Mathematics, Ewha Womans University
Seoul, 120-750, Korea
hsl@ewha.ac.kr

Abstract. The Weil and Tate pairings have been used recently to build new schemes in cryptography. It is known that the Weil pairing takes longer than twice the running time of the Tate pairing. Hence it is necessary to develop more efficient implementations of the Tate pairing for the practical application of pairing based cryptosystems. In 2002, Barreto et al. and Galbraith et al. provided new algorithms for the fast computation of the Tate pairing in characteristic three. In this paper, we give a closed formula for the Tate pairing on the hyperelliptic curve $y^2 = x^p - x + d$ in characteristic p . This result improves the implementations in [BKLS02], [GHS02] for the special case $p = 3$.

1 Introduction

Pairings were first used in cryptography as a cryptanalytic tool for reducing the discrete log problem on some elliptic curves to the discrete log problem in a finite field. There are two reduction types. One uses the Weil pairing and is called the MOV reduction [MOV93], the other uses the Tate pairing and is called the FR reduction [FR94]. Positive cryptographic applications based on pairings arose from the work of Joux [J00], who gave a simple one round tripartite Diffie-Hellman protocol on supersingular curves. Curve based pairings, such as the Weil pairing and Tate pairing, provide a good setting for the so-called bilinear Diffie-Hellman problem. Many cryptographic schemes based on the pairings have been developed recently, such as identity based encryption [BF01], identity based signature schemes [SOK00], [CC03], [H02a], [P02], and identity based authenticated key agreement [S02]. For the practical application of those systems it is important to have efficient implementations of the pairings. According to [G01], the Tate pairing can be computed more efficiently than the Weil pairing. The recent papers [BKLS02], [GHS02] provide fast computations of the Tate pairing in characteristic three.

Our main result in this paper is a closed expression for the Tate pairing on the hyperelliptic curve defined by the equation $C^d/k : y^2 = x^p - x + d$, for a

* Supported by Korea Research Foundation Grant (KRF-2002-070-C00010)

prime number p congruent to 3 modulo 4 (Theorem 5). We assume that k is a finite extension of degree n of the prime field F_p with n coprime to $2p$. The formula assigns to a pair (P, Q) of k -rational points on the curve an element $\{P, Q\} \in K^*$, where K/k is an extension of degree $2p$. By a general property of the Tate pairing the map is bilinear. Following Joux [J00], we can use the map to construct a tripartite key agreement protocol: If A, B, C are three parties with private keys a, b, c , and public keys aP, bP, cP , respectively, they can establish a common secret key $\alpha \in K^*$ via

$$\alpha = \{aP, bP\}^c = \{bP, cP\}^a = \{cP, aP\}^b \in K^*.$$

The computation of the Tate pairing can be performed using an algorithm first presented by Miller [M86]. For a general elliptic curve in characteristic three, the computation can be improved. For the elliptic curve $E^b/k : y^2 = x^3 - x + b$, techniques specific to the curve yield further improvements [BKLS02], [GHS02]. We describe these algorithms and we show that the evaluation of our closed expression, for the special case $p = 3$, uses fewer logical and arithmetic operations.

This paper is organized as follows. In the next section, we recall the general formulation of the Tate pairing. Section 3 gives useful properties of the elliptic curve $E^b : y^2 = x^3 - x + b$ and gives Miller's algorithm in base 3. We also describe the algorithm for computing the Tate pairing due to Barreto et al. [BKLS02]. For comparison, we derive a closed expression for the output of the algorithm proposed by Barreto et al. in Section 4. Section 5 gives useful properties of the curve $C^d : y^2 = x^p - x + d$ and we give a first algorithm to evaluate the Tate pairing for the curve C^d . Our main result in Section 6 gives the output of this algorithm in closed form. The expression is then used to formulate a second faster algorithm.

2 Tate Pairing

Let X/k be an algebraic curve over a finite field k . Let **Div** be the group of divisors on X , **Div**₀ the subgroup of divisors of degree zero, **Prin** the subgroup of principal divisors, and $\Gamma = \mathbf{Div}_0/\mathbf{Prin}$ the group of divisor classes of degree zero. For $m > 0$ prime to $\text{char } k$, let

$$\Gamma[m] = \{[D] \in \Gamma : mD \text{ is principal}\}.$$

For a rational function f and a divisor $E = \sum n_P P$ with $(f) \cap E = \emptyset$, let

$$f(E) = \prod f(P)^{n_P} \in k^*.$$

Theorem 1 ([FR94], [H02b]). *The Tate pairing*

$$\begin{aligned} \{-, -\}_m : \Gamma[m] \times \Gamma/m\Gamma &\longrightarrow k^*/k^{*m}, \\ \{[D], [E]\}_m &= f_D(E), \end{aligned}$$

is well-defined on divisor classes. The pairing is non-degenerate if and only if the constant field k of X contains the m -th roots of unity. Here, f_D is such that $(f_D) = mD$, and we assume that the classes are represented by divisors with disjoint support: $D \cap E = \emptyset$.

For an elliptic curve E/k we can identify Γ with the group of rational points on the curve using an isomorphism $E(k) \simeq \Gamma$, $P \mapsto [P - O]$. For an elliptic curve E/k , and for $D = [P - O]$, efficient computation of $f_D(Q)$ in the Tate pairing is achieved with a square-and-multiply strategy using Miller's algorithm in base 2 [M86].

3 The BKLS-Algorithm

Let $E^+ : y^2 = x^3 - x + 1$ and $E^- : y^2 = x^3 - x - 1$ be twisted elliptic curves over the field F_3 of three elements. Their cryptographic applications have been studied in [K98], [DS98]. Let N be the number of points on E^+ or E^- over an extension field $k = F_{3^n}$ such that $\gcd(n, 6) = 1$. Then the Tate pairing

$$\begin{aligned} \{-, -\}_N : \Gamma[N] \times \Gamma/N\Gamma &\longrightarrow K^*/K^{*N}, \\ \{[D], [E]\}_N &= f_D(E), \end{aligned}$$

is non-degenerate for an extension K/k of degree $[K : k] = 6$. For the extension K/k , $E(K)$ contains the full N -torsion and the Weil pairing is also non-degenerate [MOV93].

For the curves E^b , $b = \pm 1$, multiplication $V \mapsto 3V$ is particularly simple. For $V = (\alpha, \beta)$, $3V = (\alpha^9 - b, -\beta^9)$. Also, taking the cube of a scalar $f \mapsto f^3$ in characteristic three has linear complexity on a normal basis. Thus, Miller's algorithm will perform faster for these curves in a cube-and-multiply version (Algorithm 1).

Next we describe further improvements to Algorithm 1 proposed in [BKLS02], [GHS02]. We consider the curve $E^b/k : y^2 = x^3 - x + b$, for $b = \pm 1$. We assume k is of finite degree $[k : F_3] = n$ with $\gcd(n, 6) = 1$. And we let F/k and K/k be extensions of degree $[F : k] = 3$ and $[K : k] = 6$, respectively. The following theorem and lemma are similar to Theorem 1 and Lemma 1, respectively, in [BKLS02].

Theorem 2. *Let $N = |E(k)|$. Let $P, O \in E(k)$ be distinct points, and let g_P be a k -rational function with $(g_P) = N(P - O)$. For all $Q \in E(K)$, $Q \neq P, O$,*

$$\{[P - O], [Q - O]\}_N^{[K^*]/N} = g_P(Q)^{[K^*]/N} \in K^*.$$

Proof. Taking a power of the Tate pairing gives a non-degenerate pairing with values in K^* instead of K^*/K^{*N} . We give a different proof to show that the point O in $Q - O$ can be ignored. Let t_O be a k -rational local parameter for O , i.e. t_O vanishes to the order one in O . We may assume that $(t_O) \cap P = \emptyset$. Thus $Q - O + (t_O) \sim Q - O$, such that $Q - O + (t_O) \cap P - O = \emptyset$. With the following lemma, $g_P(Q - O + (t_O)) = g_P(Q) \in K^*/K^{*N}$. \square

Algorithm 1 Miller's algorithm, cube-and-multiply [GHS02], [BKLS02].

INPUT: $P, Q \in E(K), (a_i) \in \{0, \pm 1\}^s$.

$$\{a = 3^s + a_1 3^{s-1} + \cdots + a_{s-1} 3 + a_s.\}$$

OUTPUT: $f_a(Q)$.

$$\{(f_a) = a(P) - (aP) - (a-1)O, (l_{A,B}) = A + B + (-A - B) - 3O.\}$$

$$a \leftarrow 1, V \leftarrow P, f \leftarrow 1$$

for $i = 1$ **to** s **do**

$$g \leftarrow l_{V,V}/l_{2V,O} \cdot l_{V,2V}/l_{3V,O}(Q)$$

$$a \leftarrow 3a, V \leftarrow 3V, f \leftarrow f^3 \cdot g$$

if $a_i = \pm 1$ **then**

$$g \leftarrow l_{\pm P,V}/l_{V \pm P,O}(Q)$$

$$a \leftarrow a \pm 1, V \leftarrow V \pm P, f \leftarrow f \cdot g$$

end if

$$\{a \leftarrow 3^i + a_1 3^{i-1} + \cdots + a_{i-1} 3 + a_i, V \leftarrow aP, f \leftarrow f_a(Q).\}$$

end for

Lemma 1. Let $N = |E(k)|$. For a F -rational function f and for a F -rational divisor R such that $(f) \cap R = \emptyset$,

$$f(R) = 1 \in K^*/K^{*N}.$$

Proof. We have $f(R) \in F^*$. The group order N is an odd divisor of $3^{3n} + 1$. Therefore, the group order N is coprime to $3^{3n} - 1$. And $F^* = F^{*N} \subset K^{*N}$. \square

Definition 1 ([BKLS02]). Let $\rho \in F_{3^3}$ be a root of $\rho^3 - \rho - b = 0$. Let $\sigma \in F_{3^2}$ be a root of $\sigma^2 + 1 = 0$. Define the distortion map

$$\phi : E(K) \rightarrow E(K), \quad \phi(x, y) = (\rho - x, \sigma y). \quad (1)$$

Combine the distortion map with Theorem 2 to obtain a pairing

$$E(k) \times E(k) \longrightarrow K^*, \quad (P, Q) \mapsto g_P(\phi(Q))^{|K^*|/N} \in K^*. \quad (2)$$

The curve $y^2 = x^3 - x + b$ has complex multiplication by -1 and the distortion map corresponds to multiplication by $\sqrt{-1}$. Indeed, ϕ is an automorphism of E ,

$$(\sigma y)^2 = -y^2 = -x^3 + x - b = (\rho - x)^3 - (\rho - x) + b.$$

And $\phi^2 = -1$. The following remark is used in Theorem 3 [BKLS02] to discard contributions of the form $l_{P,O}(\phi(Q))$ in the evaluation of the Tate pairing.

Remark 1. Let $P \in E(k)$, $Q \in F \times K$, and let $l_{P,O}$ be the vertical line through P . Then $l_{P,O}(\phi(Q)) = 1 \in K^*/K^{*N}$.

Algorithm 2 $E/k : y^2 = x^3 - x + b$ [BKLS02].

INPUT: $P \in E(k), Q = (x, y) \in F \times K, a = 3^{2m-1} \pm 3^m + 1.$

$\{[k : F_3] = 2m - 1, [F : k] = 3, [K : k] = 6, a = |E(k)|.\}$

OUTPUT: $f_a(Q).$

$\{(f_a) = a(P) - (aP) - (a - 1)O, (l_{A,B}) = A + B + (-A - B) - 3O.\}$

$V \leftarrow P, a \leftarrow 1, f \leftarrow 1$

for $i = 1$ to $m - 1$ **do**

$g \leftarrow l_{V,V}l_{V,-3V}(Q)$

$a \leftarrow 3a, V \leftarrow 3V, f \leftarrow f^3 \cdot g \{a = 3, \dots, 3^{m-1}\}$

end for

$g \leftarrow l_{\pm P,V}(Q)$

$a \leftarrow a \pm 1, V \leftarrow V \pm P, f \leftarrow f \cdot g \{a = 3^{m-1} \pm 1\}$

for $i = 1$ to m **do**

$g \leftarrow l_{V,V}l_{V,-3V}(Q)$

$a \leftarrow 3a, V \leftarrow 3V, f \leftarrow f^3 \cdot g \{a = 3^m + 3, \dots, 3^{2m-1} \pm 3^m\}$

end for

$g \leftarrow l_{P,V}(Q)$

$a \leftarrow a + 1, V \leftarrow V + P, f \leftarrow f \cdot g \{a = 3^{2m-1} \pm 3^m + 1\}$

We summarize the differences between Algorithm 1 and Algorithm 2.

1. The distortion map gives a non-degenerate pairing on $E(k) \times E(k)$.
2. Because of the simple ternary expansion of N , a single loop of length $2m - 1$ containing an if statement for the adding can be replaced with two smaller loops each followed by an unconditional addition.
3. The denominators in $l_{V,V}/l_{2V,O} \cdot l_{V,2V}/l_{3V,O}$ are omitted. For $P \in E(k), x_Q \in F$, they do not affect the value of the Tate pairing.
4. The line $l_{V,2V}$ is written $l_{V,-3V}$. Since the points $V, 2V$ and $-3V$ lie on a line, the expressions are the same, but $-3V$ is easier to compute than $2V$. For $V = (\alpha, \beta)$, $-3V = (\alpha^9 - b, \beta^9)$.

We give a further analysis of Algorithm 2 in the following section.

4 A Closed Formula for the BKLS-Algorithm

Let $E^b/k : y^2 = x^3 - x + b$ be an elliptic curve as in Section 3. Recall from Definition 1 in Section 3 the pairing $E(k) \times E(k) \rightarrow K^*$,

$$(P, Q) \mapsto g_P(\phi(Q))^{K^*/N} \in K^*.$$

For the efficient evaluation of $g_P(\phi(Q))$ we use Algorithm 2.

Remark 2. We make three remarks. They all reflect that the lines that are computed by the algorithm can be precomputed.

1. After the first loop, we have, for $P = (\alpha^3, \beta^3)$,

$$l_{\pm P, V} = \pm y - \beta(x - \alpha + b).$$

2. After the second loop $V = (3^{2m-1} \pm 3^m)P = -P$, and multiplication by $l_{P, -P}(Q) = l_{P, 0}(Q)$ can be omitted.
3. Inside each loop, if we omit only the denominator $l_{3V, O}$, we find

$$(l_{V, V} l_{V, -3V} / l_{2V, O}) = 3V + (-3V) - 4O.$$

For $V = (\alpha, \beta)$, the function $h_V : \beta^3 y - (\alpha^3 - x + 1)^2$ has the same divisor. We claim that using h_V in place of $l_{V, V} l_{V, -3V}$ uses fewer operations.

Theorem 3 (Algorithm 2 in closed form). *Let*

$$P = (\alpha^3, \beta^3) \in E(k), \quad Q = (x, y) \in E(k), \quad \phi(Q) = (\rho - x, \sigma y).$$

Then, for g_P with $(g_P) = N(P - Q)$, $g_P(\phi(Q))$ is the product of

$$\begin{aligned} & \prod_{i=1}^{m-1} (\sigma \beta^{(i)} y^{(n-i)} - (\alpha^{(i)} + x^{(n-i)} - \rho + mb)^2), \\ & \prod_{i=m}^{2m-1} (\sigma \beta^{(i)} y^{(n-i)} - (\alpha^{(i)} + x^{(n-i)} - \rho)^2), \\ & (\pm \sigma y - \beta(\rho - x - \alpha + b))^{(m)}. \end{aligned}$$

The second remark is clear. In the remainder of this section we first prove the third remark, then the first remark and finally the theorem.

Lemma 2. *Let $l_{A, B}$ be the line through A and B . For $V = (\alpha, \beta) \in E(K)$,*

$$\begin{aligned} l_{V, V} &: (x - \alpha) - \beta(y - \beta) = 0, \\ l_{2V, O} &: x - \alpha - 1/\beta^2 = 0, \\ l_{2V, V} &: (\beta^4 - 1)(x - \alpha) - \beta(y - \beta) = 0, \\ l_{3V, O} &: x - \alpha^9 + b = 0. \end{aligned}$$

The lines $l_{V, V}, l_{2V, V}$ correspond to l_1 and l'_1 , respectively, in [GHS02], up to a slight difference to reduce the number of operations. For the third remark, we compare the number of operations (Multiplication, Squaring, Addition, Frobenius).

$$\begin{aligned} g &\leftarrow l_{V, V} l_{V, -3V}, f \leftarrow f^3 \cdot g & (4M, 4A, 1F) \\ g &\leftarrow h_V, f \leftarrow f^3 \cdot g & (2M, 1S, 2A, 2F) \end{aligned}$$

To establish the first remark we use the following lemma.

Lemma 3. *Let $(\alpha, \beta) \in E^b(\bar{F}_3)$. The line $l : by - \beta(x - \alpha + b) = 0$ has divisor*

$$(\alpha, \beta) + (\alpha + b, -\beta) + (\alpha^3, b\beta^3) - 3O.$$

Let $(\alpha, \beta) \in E^b(k)$, for k of degree $[k : F_3] = n = 2m - 1$ with $\gcd(6, n) = 1$.

$$n = 1(\bmod 3) : \quad 3^n(\alpha + b, -\beta) = (\alpha, \beta), \quad 3^m(\alpha + b, -\beta) = (\alpha^3, (-1)^{m+1}\beta).$$

$$n = 2(\bmod 3) : \quad 3^n(\alpha, \beta) = (\alpha + b, -\beta), \quad 3^m(\alpha, \beta) = (\alpha^3, (-1)^m\beta).$$

Proof. The first claim is obvious. The last claim uses

$$V = (\alpha, \beta) \Rightarrow 3V = (\alpha^9 - b, -\beta^9).$$

□

We summarize in a table.

	$n = 1(\bmod 3), m = 1(\bmod 3)$	$n = 2(\bmod 3), m = 0(\bmod 3)$
(α, β)	$3^n W$	W
$(\alpha + b, -\beta)$	W	$3^n W$
$(\alpha^3, b\beta^3)$	$\epsilon 3^m W$	$\epsilon 3^m W$
ϵ	$(-1)^{m-1}b$	$(-1)^m b$

With the value of ϵ from the table, $|E(k)| = 3^n + 1 + \epsilon 3^m$.

Proposition 1. *We apply the lemma. Let $P = (\alpha^3, \beta^3) \in E^b(k)$, for k of degree $[k : F_3] = n = 2m - 1$ with $\gcd(6, n) = 1$. The line through ϵP and $V = 3^{m-1}P$ has equation*

$$l_{\epsilon P, V} : \epsilon y - \beta(x - \alpha + b) = 0.$$

The third point on the line $l_{\epsilon P, V}$ is $(\alpha + mb, (-1)^m \beta)$.

Proof. Write $P = 3^m W$, so that $V = 3^n W$. Then W is the third point on the line through ϵP and V . And W can be obtained as the unique solution to $3^m W = P$. □

This proves the first remark. We can now prove Theorem 3.

Proof. The contribution of the first loop to $g_P(\phi(Q))$ is

$$\begin{aligned}
 & \prod_{i=1}^{m-1} ((-1)^{i-1} \beta^{(2i)} (\sigma y) - (\alpha^{(2i)} - (i-1)b - (\rho - x) + b)^2)^{(2m-1-i)} \\
 &= \prod_{i=1}^{m-1} ((-1)^{i-1} \beta^{(i)} (\sigma^{(n-i)} y^{(n-i)} \\
 & \quad - (\alpha^{(i)} - (i-1)b - (\rho + (2m-1-i)b - x^{(n-i)} + b)^2) \\
 &= \prod_{i=1}^{m-1} (\beta^{(i)} y^{(n-i)} \sigma - (\alpha^{(i)} + x^{(n-i)} - \rho + mb)^2).
 \end{aligned}$$

The second loop starts with $V = (\alpha + mb, (-1)^m \beta)$ instead of $V = P = (\alpha^3, \beta^3)$ and is of length m instead of length $m - 1$. It gives a contribution

$$\begin{aligned}
& \prod_{i=1}^m ((-1)^{i+m} \beta^{(2i-1)} (\sigma y) - (\alpha^{(2i-1)} + (m+1-i)b - (\rho-x) + b)^2)^{(m-i)} \\
&= \prod_{i=1}^m ((-1)^{i+m} \beta^{(m-1+i)} \sigma^{(m-i)} y^{(m-i)} \\
&\quad - (\alpha^{(m-1+i)} + (m+1-i)b - (\rho + (m-i)b - x^{(m-i)} + b)^2) \\
&= \prod_{i=m}^{2m-1} (\beta^{(i)} y^{(n-i)} \sigma - (\alpha^{(i)} + x^{(n-i)} - \rho - b)^2).
\end{aligned}$$

The contribution from $l_{eP,V}$ follows directly from the proposition 1. This proves Theorem 3. \square

5 The Curve $C^d : y^2 = x^p - x + d$

Let C^d/k be the hyperelliptic curve $y^2 = x^p - x + d$, $d = \pm 1$, for $p \equiv 3 \pmod{4}$. We assume that k is of degree $[k : F_p] = n$, for $\gcd(2p, n) = 1$, and we let F/k and K/k be the extensions of degree $[F : k] = p$ and degree $[K : k] = 2p$, respectively. Thus C^d is a direct generalization of the elliptic curve E^b studied in the previous sections. Over the extension field K , the curve is the quotient of a hermitian curve, hence is Hasse-Weil maximal. And the class group over K is annihilated by $p^{pn} + 1$. The last fact can be seen also from the following lemma. It shows that for $P \in C^d(K)$, $(p^{pn} + 1)(P - O)$ is principal. We write $x^{(i)}$ for x^{p^i} .

Lemma 4 ([D96],[DS98]). *Let $P = (\alpha, \beta) \in C^d$. The function*

$$h_P = \beta^p y - (\alpha^p - x + d)^{(p+1)/2}$$

has divisor $(h_V) = p(V) + (V') - (p+1)O$, where

$$V' = (\alpha^{(2)} + d^p + d, \beta^{(2)}).$$

We will write V also for the divisor class $V - O$, so that $V' = -pV$. In particular $p^{pn}P = -P$, for $P \in C(K)$ and for $\text{Trace}_{K/F_p} d = 0$. Let $M = p^{pn} + 1 = |K^*|/|F^*|$. Thus, the order of $P - O$ in the divisor class group Γ is a divisor of M . The precise order N of the class group can be obtained from the zeta functions for C^d in [D96], [DS98]. We will only need the following lemma.

Lemma 5. *Let Γ^d denote the class group of the curve C^d/k .*

$$|\Gamma^+(k)| |\Gamma^-(k)| = (p^{pn} + 1)/(p^n + 1)$$

In particular, $N = |\Gamma(k)|$ is an odd divisor of $M = p^{pn} + 1$.

We include the size of the class group for $p = 7$. Let $[k : F_7] = n$ and $m = (n+1)/2$. Then

$$|\Gamma^+(k)| = (1 + 7^n)^3 + \left(\frac{7}{n}\right)7^m(1 + 7^n + 7^{2n}).$$

$$|\Gamma^-(k)| = (1 + 7^n)^3 - \left(\frac{7}{n}\right)7^m(1 + 7^n + 7^{2n}).$$

And $|\Gamma^+(k)||\Gamma^-(k)| = (1 + 7^{7n})/(1 + 7^n)$.

6 Main Theorem

Miller's algorithm for the Tate pairing on an elliptic curve E/k uses lines as building blocks to construct other rational functions. In our version of the Tate pairing implementation, we will not rely on lines but on the functions described in Lemma 4. So that we can generalize from elliptic curves $E^b/k : y^2 = x^3 - x + b$, $b = \pm 1$, to hyperelliptic curves $C^d/k : y^2 = x^p - x + d$, $d = \pm 1$, for $p \equiv 3 \pmod{4}$. Generalization of the results in Section 3 poses no problem.

Theorem 4. *Let $N = |\Gamma(k)|$, so that N divides $M = p^{pn} + 1 = |K^*|/|F^*|$. Let $P, O \in C(k)$ be distinct points. Let f_P be a k -rational function with $(f_P) = M(P - O)$. For all $Q \in C(K)$, $Q \neq P, O$,*

$$\{[P - O], [Q - O]\}_M^{|K^*|/M} = f_P(Q)^{|F^*|} \in K^*.$$

Proof. The argument that shows that the contribution by O can be omitted is the same as in Theorem 2. \square

The difference with Theorem 2 is that f_P is computed with a multiple M of N instead of with N itself. The multiple M has trivial expansion in base p and this leads to Algorithm 3 which has no logical decisions (only point multiplication by p and no adding). See also Remark in Section 6 of [GHS02]. But it has pn iterations compared to n iterations in Algorithm 2 (for the case $p = 3$). After Theorem 5, we will reduce this to n iterations in Algorithm 4. The following generalizations of Lemma 1 and Remark 1 are straightforward.

Lemma 6. *Let $N = |\Gamma(k)|$. For a F -rational function f and for a F -rational divisor E such that $(f) \cap E = \emptyset$,*

$$f(E) = 1 \in K^*/K^{*N}.$$

Proof. We have $f(E) \in F^*$. The group order N is an odd divisor of $p^{pn} + 1$. Therefore, the group order N is coprime to $p^{pn} - 1$. And $F^* = F^{*N} \subset K^{*N}$. \square

Remark 3. Let $P \in E(F)$, $Q \in F \times K$, and let $l_{P,O}$ be the vertical line through P . Then $l_{P,O}(\phi(Q)) = 1 \in K^*/K^{*N}$.

Definition 2. *Let $\rho \in F$ be a root of $\rho^p - \rho + 2d = 0$. Let $\sigma, \bar{\sigma} \in K$ be the roots of $\sigma^2 + 1 = 0$. Define the distortion map*

$$\phi : C(K) \rightarrow C(K), \quad \phi(x, y) = (\rho - x, \sigma y). \quad (3)$$

Combine the distortion map with Theorem 4 to obtain a pairing

$$C(k) \times C(k) \longrightarrow K^*, \quad (P, Q) \mapsto f_P(\phi(Q))^{|F^*|} \in K^*. \quad (4)$$

Algorithm 3 $C/k : y^2 = x^p - x + d$.

INPUT: $P \in C(k), Q \in C(K), a = p^{pn} + 1$

$$\{[k : F_p] = n, [K : k] = 2p, a = |K^*|/|F^*|.\}$$

OUTPUT: $f_a(Q) \in K^*/F^*$

$$\{(f_a) = a(P) - (aP) - (a-1)O, (h_V) = p(V) + (-pV) - (p+1)O.\}$$

$$V \leftarrow P, a \leftarrow 1, n \leftarrow 1, d \leftarrow 1$$

for $i = 1$ **to** pn **do**

$$g \leftarrow h_V(Q)$$

$$a \leftarrow pa, V \leftarrow pV, f \leftarrow f^p \cdot g$$

end for

$$\text{Indeed, } (\sigma v)^2 = -v^2 = -u^p + u - d = (\rho - u)^p - (\rho - u) + d.$$

Theorem 5 (Main Theorem). For $P = (\alpha, \beta), Q = (x, y) \in C(k)$,

$$f_P(\phi(Q)) = \prod_{i=1}^n (\beta^{(i)} y^{(n+1-i)} \bar{\sigma} - (\alpha^{(i)} + x^{(n+1-i)} - \rho + d)^{(p+1)/2}).$$

Proof. From Algorithm 3, we see that

$$f_P(\phi(Q)) = \prod_{i=1}^{pn} (h_{p^{i-1}P}(\phi(Q))^{(pn-i)}).$$

Substitution of

$$\begin{aligned} h_P(Q) &= \beta^p y - (\alpha^p - x + d)^{(p+1)/2} \\ p^{i-1}P &= (\alpha^{(2i-2)} + (i-1)2d, (-1)^{i-1}\beta^{(2i-2)}) \\ \phi(Q) &= (\rho - x, \sigma y) \end{aligned}$$

yields

$$\begin{aligned} & \prod_{i=1}^{pn} ((-1)^{i-1} \beta^{(2i-1)} (\sigma y) - (\alpha^{(2i-1)} + (i-1)2d - (\rho - x) + d)^{(p+1)/2})^{(pn-i)} \\ &= \prod_{i=1}^{pn} ((-1)^{i-1} \beta^{(i-1)} \sigma^{(pn-i)} y^{(pn-i)} \\ & \quad - (\alpha^{(i-1)} + (i-1)2d - (\rho - (pn-i)2d - x^{(pn-i)}) + d)^{(p+1)/2}). \end{aligned}$$

Or, since $\alpha, \beta, x, y \in k$, and since $(-1)^{i-1} \sigma^{(pn-i)} = \sigma$, for both i odd and i even,

$$\begin{aligned} & \prod_{i=1}^n (\beta^{(i-1)} y^{(n-i)} \sigma - (\alpha^{(i-1)} - \rho + x^{(n-i)} - d)^{(p+1)/2})^p \\ &= \prod_{i=1}^n (\beta^{(i)} y^{(n+1-i)} \bar{\sigma} - (\alpha^{(i)} + x^{(n+1-i)} - \rho^p - d)^{(p+1)/2}). \end{aligned}$$

Finally, $-\rho^p - d = -\rho + d$.

□

Note that $f_P(\phi(Q)) = f_Q(\phi(P))$, as it should.

Algorithm 4 $C/k : y^2 = x^p - x + d$.

INPUT: $P = (\alpha, \beta) \in C(k)$, $Q = (\rho - x, \sigma y)$, $(x, y) \in C(k)$, $a = p^{pn} + 1$

$$\{[k : F_p] = n, \rho^p - \rho + 2d = 0, \sigma^2 + 1 = 0.\}$$

$$\{[F : F_p] = pn, [K : F_n] = 2pn, a = |K^*|/|F^*|.\}$$

OUTPUT: $f_a(Q) \in K^*/F^*$

$$\{(f_a) = a(P) - (aP) - (a-1)O.\}$$

for $i = 1$ **to** n **do**

$$\alpha \leftarrow \alpha^3, \beta \leftarrow \beta^3$$

$$g \leftarrow (\beta y \bar{\sigma} - (\alpha + x - \rho + d)^{(p+1)/2})$$

$$f \leftarrow f \cdot g$$

$$x \leftarrow x^{1/3}, y \leftarrow y^{1/3}$$

end for

Summarizing, using a Tate pairing $\{-, -\}_M$ instead of $\{-, -\}_N$ removes all logic and all additions from Algorithm 2. When using the version Algorithm 4 the number of iterations is similar to Algorithm 2. Which gives the following advantages for Algorithm 4.

1. Uniform algorithm that applies to all $p \equiv 3 \pmod{4}$.
2. Expressing $N = |F(k)|$ in base p can be omitted.
3. Expressing $|K^*|/N$ in base p , for raising $g_P(Q)$ to the power $|K^*|/N$, can be omitted. It is replaced with raising to the power $|F^*|$.
4. At each iteration, only multiplication by p is required, no additions.
5. Multiplication by p using the function h_P is faster than using a product of lines (case $p = 3$).

7 Concluding Remarks

Theorem 3 for elliptic curves and its generalization Theorem 5 for hyperelliptic curves give closed formulae to evaluate the Tate pairing on curves of the form $y^2 = x^p - x + d$. The complexity estimate after Lemma 3 indicates a speed-up by a factor two over algorithms described in [BKLS02] and [GHS02] when using Theorem 3 to evaluate the Tate pairing. Timing comparisons by Keith Harrison confirm this estimate. A running time comparison for the closed formula for hyperelliptic curves remains to be done. We thank Steven Galbraith, Paulo Barreto, Doug Kuhlman, Keith Harrison and anonymous referees for their helpful feedback on the preprint version.

References

- PBCL. The Pairing-Based Crypto Lounge, <http://planeta.terra.com.br/informatica/paulobarreto/pblounge.html>

- BKLS02. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems." *Advances in Cryptology – Crypto 2002*, Lecture Notes in Computer Science, Vol. 2442, Springer-Verlag, pp. 354–368, (2002).
- BSS99. I. Blake, G. Seroussi, and N.P. Smart, *Elliptic curves in cryptography*. London Mathematical Society LNS, 265. Cambridge University Press, Cambridge, 1999 (reprinted 2000).
- BF01. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing." *Advances in Cryptology, Crypto 2001*, Lecture Notes in Computer Science, Vol. 2139, Springer-Verlag, pp. 213–229, (2001).
- BS02. D. Boneh and A. Silverberg, "Applications of multilinear forms to cryptography." *Contemporary Mathematics*, Vol. 324, American Mathematical Society, pp. 71–90, (2003).
- CC03. J. C. Cha, J. H. Cheon, "An Identity-Based Signature from Gap Diffie-Hellman Groups." *Proceedings of PKC*, Lecture Notes in Computer Science, Vol. 2567, pp. 18–30, (2003).
- DH76. W. Diffie and M. Hellman. "New direction in cryptography." *IEEE Trans. Information Theory*, IT-22(6), pp. 644–654, (1976).
- D96. I. Duursma, "Class numbers for hyperelliptic curves." In: "Arithmetic, Geometry and Coding Theory." eds. Pellikaan, Perret, Vladuts, pp. 45–52, publ. deGruyter, Berlin, 1996.
- DS98. I. Duursma, K. Sakurai, "Efficient algorithms for the Jacobian variety of hyperelliptic curves $y^2 = x^p - x + 1$ over a finite field of odd characteristic p ." *Coding theory, cryptography and related areas (Guanajuato, 1998)*, pp. 73–89, Springer, Berlin, 2000.
- FR94. G. Frey, H.-G. Rück, "A remark concerning m -divisibility and the discrete logarithm in the divisor class group of curves." *Math. Comp.* 62, no. 206, pp. 865–874, (1994).
- G01. S.D. Galbraith, "Supersingular curves in cryptography." *Asiacrypt 2001*, Springer, Lecture Notes in Computer Science, Vol. 2248, 495–513, (2001).
- GHS02. S. D. Galbraith, K. Harrison, D. Soldera, "Implementing the Tate pairing." *Algorithmic Number Theory Symposium, ANTS-V*, Lecture Notes in Computer Science, Vol. 2369, Springer-Verlag, pp. 324–337, (2002).
- H02a. F. Hess, Exponent group signature schemes and efficient identity based signature schemes based on pairing, *Proceedings of the Workshop Selected Areas in Cryptology, SAC*, Aug. 2002.
- H02b. F. Hess, "A Note on the Tate Pairing of Curves over Finite Fields," 2002. Available on <http://www.math.tu-berlin.de/~hess>.
- IT02. T. Izu and T. Takagi, "Efficient Computations of the Tate Pairing for the Large MOV degrees." *5th International Conference on Information Security and Cryptology, ICISC 2002*, Springer-Verlag, Lecture Notes in Computer Science, Vol. 2587, pp. 283–297, (2003).
- J00. A. Joux, "A one round protocol for tripartite Diffie-Hellman." *Proceedings of Algorithmic Number Theory Symposium*, Lecture Notes in Computer Science, Vol. 1838, Springer-Verlag, pp. 385–394, (2000).
- K98. N. Koblitz, "An elliptic curve implementation of the finite field digital signature algorithm." *Advances in cryptology, Crypto 1998*, Lecture Notes in Computer Science, Vol. 1462, Springer, Berlin, pp. 327–337, 1998.
- M86. V. Miller, "Short Programs for Functions on Curves." Unpublished manuscript, 1986.

- MOV93. A. Menezes, T. Okamoto and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field." IEEE Trans. on Inform. Theory 39, pp. 1639–1646, (1993).
- P02. K.G. Paterson, ID-based signature from pairings on elliptic curves, Electronics Letters, Vol. 38 (18), pp. 1025-1026, (2002).
- SOK00. R. Sakai, K. Ohgishi and M. Kasahara, "Cryptosystems based on pairing." Symposium on cryptography and Information Security, Okinawa, Japan, pp. 26-28, (2000)
- S02. N.P. Smart, An identity based authentication key agreement protocol based on pairing, Electronics Letters, Vol 38, pp 630-632, (2002).

The AGM- $X_0(N)$ Heegner Point Lifting Algorithm and Elliptic Curve Point Counting

David R. Kohel

School of Mathematics and Statistics
University of Sydney, NSW 2006, Australia
kohel@maths.usyd.edu.au

Abstract. We describe an algorithm, AGM- $X_0(N)$, for point counting on elliptic curves of small characteristic p using p -adic lifts of their invariants associated to modular curves $X_0(N)$. The algorithm generalizes the construction of Satoh [10], SST [11], and Mestre [9]. We describe this method and give details of its implementation for characteristics 2, 3, 5, 7, and 13.

Keywords: Elliptic curve cryptography, modular curves, point counting

1 Introduction

Elliptic curve cryptosystems can be designed using the reduction of precomputed CM curves or using randomly selected curves over a finite field. In the former case, the curve can be assumed to be drawn from a prespecified list of curves having many endomorphisms, on which an adversary can perform precomputations or exploit the existence of endomorphisms of small degree. On the general randomly selected curve, the only endomorphisms of small degree are scalar multiplication by a small integer. Such curves are believed to have higher security, but to implement an elliptic curve cryptosystem using randomly generated curves, it is imperative to have an efficient algorithm to determine the number of points on arbitrary elliptic curves.

The first theoretically polynomial time algorithm for point counting was due to Schoof [13]. Atkin and Elkies (see [3]) introduced the use of modular parametrizations of the torsion subgroups of elliptic curves to turn Schoof's algorithm into a practical one. Couveignes introduced an extension of this algorithm to curves over finite fields of small characteristic, and independently Lercier designed an efficient algorithm specific to characteristic 2.

In 1999, Satoh [10] introduced a novel idea of p -adically lifting the j -invariants of the cycle of curves which are related by the Frobenius isogeny $(x, y) \mapsto (x^p, y^p)$ over a finite field $\mathbb{F}_q = \mathbb{F}_{p^m}$ of small characteristic p . The j -invariants $j_0, j_1, \dots, j_m = j_0$ can be lifted efficiently to a degree m extension of the p -adic field \mathbb{Q}_p even though to lift the j -invariants to an extension of \mathbb{Q} would in general require an extension of degree $O(\sqrt{q})$. The classical modular polynomial $\Phi_p(X, Y)$ provides the algebraic lifting condition. The unique p -adic lifts \tilde{j}_i are those for which the equations $\Phi_p(\tilde{j}_i, \tilde{j}_{i+1}) = 0$ continue to hold. This was followed

by the exposition of extensions to characteristic 2 in [4] and [11]. Subsequently, in 2001, Mestre [9] introduced the use of the arithmetic–geometric mean, or AGM, to obtain elementary convergent recursion relations for the invariants of the p -adic lift of an elliptic curve.

In this work, we introduce a family of algorithms AGM- $X_0(N)$ given by convergent p -adic recursions for determining the p -adic lifts of *Heegner points* on modular curves $X_0(N)$. Heegner points are special points on modular curves which correspond to exceptional elliptic curves with CM, and are invariants from which we can “read off” the data for the trace of Frobenius, determining its number of points over \mathbb{F}_q . Specifically, we describe how the univariate version of Mestre’s method as described in Gaudry [5] and Satoh [12] relates to the AGM- $X_0(8)$, and present essentially new generalizations AGM- $X_0(2)$, AGM- $X_0(4)$, and AGM- $X_0(16)$ which apply to point counting on elliptic curves in characteristic 2. In general this method is applicable to point counting on elliptic curves of any small characteristic p , with complete details described here for characteristics 2, 3, 5, 7, and 13.

The present work creates a general framework for point counting on elliptic curves over fields of small characteristic. While the AGM point counting method for even characteristic fields had outpaced comparable algorithms for curves over fields of other small characteristics, as well as the SEA for prime fields, the present AGM- $X_0(N)$ variants of the algorithm place all small characteristic base fields on an equal footing. Exploitation of the AGM for cryptographic constructions or any potential cryptanalytic attacks should therefore extend naturally to any small characteristic base field. The main elliptic curve standards admit only extensions of the binary field or large prime fields, but the omission of odd characteristic extension fields is not based on security considerations. Cryptographic standards for odd characteristic extension fields have been proposed [6], in part to permit efficient software implementations of curves over medium-sized characteristic [1]. A generic framework for odd characteristic extension fields also applies to fields of small characteristic, and makes it imperative to advance the theory of applicable algorithms and cryptographic characteristics of elliptic curves over arbitrary finite fields.

2 Modular Curves and Parametrizations

A modular curve $X_0(N)$ parametrizes elliptic curves together with some cyclic N -torsion subgroup. The simplest case is the modular curve $X_0(1)$ which classifies elliptic curves up to isomorphism via their j -invariants. Associated to any j other than 0 or 12^3 , we can write down a curve

$$E : y^2 + xy = x^3 - \frac{36}{j - 12^3}x - \frac{1}{j - 12^3}$$

with associated invariant j . The curve $X_0(1)$ is identified with the line of j -values, each point corresponding to the class of curves with invariant j . The next simplest case is the curve $X_0(2)$, which is described by a function s_1 , and which classifies an elliptic curve together with a 2-torsion subgroup.

$$E_1 : y^2 + xy = x^3 - 128s_1x^2 - \frac{36s_1}{64s_1 + 1}x + \frac{512s_1^2 - s_1}{64s_1 + 1}.$$

The j -invariant of this curve is $j = (256s_1 + 1)^3/s_1$ and the 2-torsion subgroup is specified by $P = (-1/4, 1/8)$. The quotient of the curve E_1 by this group gives a new curve

$$F_1 : y^2 + xy = x^3 - 128s_1x^2 - \frac{327680s_1^2 + 3136s_1 + 5}{16(64s_1 + 1)}x + \frac{(512s_1 + 1)(262144s_1^2 + 1984s_1 + 3)}{64(64s_1 + 1)},$$

with j -invariant $(16s_1 + 1)^3/s_1^2$. If we try to put the curve F_1 into the form

$$E_2 : y^2 + xy = x^3 - 128s_2x^2 - \frac{36s_2}{64s_2 + 1}x + \frac{512s_2^2 - s_2}{64s_2 + 1}.$$

for some s_2 , then we necessarily have an equality of their j -invariants

$$j(F_1) = (16s_1 + 1)^3/s_1^2 = (256s_2 + 1)^3/s_2 = j(E_2),$$

which gives rise to an relation $s_1^2 - 4096s_1s_2^2 - 48s_1s_2 - s_2 = 0$ between the s -invariants on E_1 and E_2 , where we discard the trivial factor $4096s_1s_2 - 1$, determining the parametrized dual isogeny

$$\begin{array}{ccc} E_1 & \xrightarrow{\phi} & E_1/\langle(0,0)\rangle = F_1 \\ \cong \uparrow & & \downarrow \cong \\ F_2 = E_2/\langle(0,0)\rangle & \xleftarrow{\hat{\phi}} & E_2. \end{array}$$

For the former equation, the resulting composition $\phi_1 : E_1 \rightarrow F_1 \cong E_2$, which may only exist over a quadratic extension of the field generated by s_1 and s_2 , can be shown to induce the pullback $\phi_1^*\omega_2 = \pi(s_1, s_2)\omega_1$ where

$$\pi(s_1, s_2) = 2 \left(\frac{(256s_1 + 1)(512s_2(64s_2 + 1) - 8s_1 + 1)}{(256s_2 + 1)(-256s_2(256s_2 + 1) + 16s_1 + 1)} \right)^{1/2},$$

and where ω_1 and ω_2 are the invariant differentials $dx/2y$ on the respective curves E_1 and E_2 . Since the reduction of the relation between the s -invariants of the curves E_1 and E_2 gives $s_2 \equiv s_1^2 \pmod{2}$, and the kernel is defined by to be those points (x, y) for which $4x - 1 \equiv 0$, we conclude that ϕ_1 defines a parametrized lift of the Frobenius isogeny.

The isogeny ϕ_1 can be extended similarly by an isogeny ϕ_2 ,

$$E_1 \longrightarrow F_1 \cong E_2 \longrightarrow F_2 \cong E_3 \longrightarrow \dots$$

and the corresponding cycle of invariants s_1, s_2, \dots, s_m , linked by a chain of isogeny relations, the product of the $\pi_i = \pi(s_i, s_{i+1})$ determines the action of Frobenius on the space of differentials of E_1 and we can read off its trace, which determines the number of points on the curve. This is the basis of the algorithm of Satoh [10] using the j -invariant and the algorithm of Mestre [9] using modular parametrizations of elliptic curves by the curve $X_0(8)$. The above example provides the equations necessary to use the curve $X_0(2)$ in an analogous manner.

2.1 Modular Correspondences

The equation $\Phi(s_1, s_2) = s_1^2 - 4096s_1s_2^2 - 48s_1s_2 - s_2 = 0$, derived in the previous section, is an example of a modular correspondence. The function s on $X_0(2)$ generates the function field, and the relation between s_1 and s_2 determines the image of the modular curve $X_0(4)$ in the product $X_0(2) \times X_0(2)$.

At a high level we extend this construction as follows. A point on a modular curve $X_0(N)$ corresponds to the isomorphism class of a point (E, G) , where E is an elliptic curve and G is a subgroup isomorphic to $\mathbb{Z}/N\mathbb{Z}$. For any such pair (E, G) we may associate the quotient curve $F = E/G$ together with the quotient isogeny $\phi : E \rightarrow F$. Conversely, to any isogeny $\phi : E \rightarrow F$ with cyclic kernel of order N , we can associate the pair $(E, \ker(\phi))$. We say that a map of curves $X_0(pN) \rightarrow X_0(N) \times X_0(N)$ is an *oriented modular correspondence* if the image of each point representing a pair (E, G) maps to $((E_1, G_1), (E_2, G_2))$ where $E_1 = E$ and G_1 is the unique subgroup of index p in G , and where $E_2 = E/H$ and $G_2 = G/H$, where H is the unique subgroup of order p . Since the composition

$$\phi : E = E_1 \rightarrow E_2 \rightarrow E_2/G_2 = E/G,$$

recovers the pair (E, G) , one considers the point (E_2, G_2) as an extension of the degree N isogeny $\phi_1 : E_1 \rightarrow E_1/G_1$ determined by (E_1, G_1) to the isogeny of degree pN determined by (E, G) . When the curve $X_0(N)$ has genus zero, there exists a single function x which generates its function field, and the correspondence can be expressed as a binary equation $\Phi(x, y) = 0$ in $X_0(N) \times X_0(N)$ cutting out $X_0(pN)$ inside of the product.

At a more basic level, the construction is determined as follows. Let $x = x(q)$ be a suitable modular function generating the function field of a genus zero curve $X_0(N)$, represented as a power series. Then $y = x(q^p)$ is a modular function on $X_0(pN)$, and an algebraic relation $\Phi(x, y) = 0$ determines an oriented modular correspondence as above. The application of modular correspondences to the lifting problem for elliptic curves is based on the following theorem.

Theorem 1. *Let p be a prime dividing N and let $\Phi(x, y) = 0$ be the equation defining an oriented modular correspondence $X_0(pN) \rightarrow X_0(N) \times X_0(N)$ on a modular curve $X_0(N)$ of genus zero such that $\Phi(x, y) \equiv y^p - x \pmod{p}$. Let $x_1, x_2, \dots, x_m, x_{m+1} = x_1$ be a sequence of $m > 2$ distinct algebraic integers in some unramified extension of \mathbb{Q}_p such that $\Phi(x_i, x_{i+1}) = 0$. Then the x_i form a Galois conjugacy class of invariants of CM curves.*

The above theorem describes the relation between cycles of points on modular curves and CM curves. A sequence of points satisfying the conditions of the theorem are examples of *Heegner points* on $X_0(N)$. After an initial precomputation to determine the equations as presented in this article, it is sufficient to dispense with the elliptic curves and compute only with their modular invariants. The defining functions and relations for the family determine the particular algorithm AGM- $X_0(N)$ to be used for p -adic lifting. Each is denoted according to the modular curve $X_0(N)$ on which we lift points. In each instance we

have an initial condition of the form $x_1 \equiv 1/j \pmod{p}$ and a recursion for computing the function x_{i+1} in terms of x_i , which arises from the correspondence $X_0(2N) \rightarrow X_0(N) \times X_0(N)$ given by the equations $\Phi(x_i, x_{i+1}) = 0$ as below.

$$\begin{array}{ll} \underline{X_0(2)} : s_1^2 - 16(256s_2 + 3)s_1s_2 - s_2 = 0, & \underline{X_0(8)} : u_1^2(4u_2 + 1)^2 - u_2 = 0, \\ \underline{X_0(4)} : t_1^2 - 16(16t_1t_2 + t_1 + t_2)t_2 - t_2 = 0, & \underline{X_0(16)} : v_1^2(4v_2^2 + 1) - v_2 = 0. \end{array}$$

The relations between the above functions are given by the identities

$$\begin{array}{ll} j_1 = (256s_1 + 1)^3/s_1, & t_1 = u_1/(-4u_1^2 + 1), \\ s_1 = t_1(16t_1 + 1), & u_1 = v_1/(1 + 4v_1^2). \end{array}$$

Each function can be expressed in terms of the classical modular functions from which their relations were derived.

Families of p -adic liftings exist for odd characteristic, and in particular, when the genus of $X_0(N)$ is zero¹ we obtain a simple relation for the correspondence $X_0(pN) \rightarrow X_0(N) \times X_0(N)$. For instance, if $p = 3$ and N is 3 or 9 we give the correspondences defining algorithms AGM- $X_0(3)$ and AGM- $X_0(9)$ below.

$$\begin{array}{l} \underline{X_0(3)} : s_1^3 - 9(59049s_1s_2^2 + 2916s_1s_2 + 81s_2 + 30s_1 + 4)s_1s_2 - s_2 = 0 \\ \underline{X_0(9)} : t_1^3 - 9((27t_1^2 + 9t_1 + 1)(3t_2 + 1)t_2 + (3t_1 + 1)t_1)t_2 - t_2 = 0 \end{array}$$

The relations between these functions and the j -invariant is given by the equations:

$$\begin{array}{l} j_1 = (27s_1 + 1)(243s_1 + 1)^3/s_1, \\ s_1 = (27t_1^2 + 9t_1 + 1)t_1. \end{array}$$

2.2 Power Series Developments

Each of the selected functions are p -adically convergent away from the supersingular point $j_1 = 0 \pmod{p}$ when $p = 2$ or 3 . The equations of the form $\Phi(x_i, x_{i+1}) = 0$ allow us to find a general solution for x_{i+1} as a power series in x_i . We note that for all functions given above, j_1^{-1} is an initial approximation to the p -adic value of x_1 .

$$\begin{array}{l} \underline{X_0(2)} : \\ s_{i+1} = s_i^2 - 48s_i^3 + 2304s_i^4 - 114688s_i^5 + 5898240s_i^6 + \dots \\ \quad = s_i^2(1 - 48s_i)(1 + 2304s_i^2)(1 - 4096s_i^3)(1 + 5701632s_i^4) \dots \\ \underline{X_0(4)} : \\ t_{i+1} = t_i^2 - 16t_i^3 + 240t_i^4 - 3584t_i^5 + 53760t_i^6 - 811008t_i^7 + \dots \\ \quad = t_i^2(1 - 16(t_i - 15t_i^2))(1 - 3584(t_i^3 + t_i^4))(1 + 13029376s_i^6)(1 - 8192s_i^5) \dots \end{array}$$

¹ The genus of $X_0(N)$ is zero if and only if N is one of the values 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 16, 18, or 25. In this case, there exists a single function which parametrizes $X_0(N)$. In the general case we would need multiple functions and the polynomial relations they satisfy. Here we will only be interested in the subset of these N which are powers of the characteristic p .

Table 1. HeegnerPointAnalyticLift.

Input: The modular polynomial $\Phi(x, y)$; the precomputed product decomposition for the analytic power series

$$y(x) = x^p f_1(x) f_2(x) \cdots \text{ such that } \Phi(x, y(x)) = 0$$

and $f_i(x) \equiv 1 \pmod{p^i}$; a finite field element x_0 such that $\Phi(x_0, x_0^p) = 0$; and a target precision w .

Output: An unramified p -adic lift x_1 of a Galois conjugate of x_0 such that (x_1, x_1^p) is a zero of Φ to precision p^w .

Set x_1 to be any p -adic lift of x_0 .

for $(1 \leq k \leq w - 1)$ {

$$x_1 = x_1^p \prod_{i=1}^k f_i(x_1) \pmod{p^{k+1}}$$

}

return x_1

$X_0(8)$:

$$\begin{aligned} u_{i+1} &= u_i^2 + 8u_i^4 + 80u_i^6 + 896u_i^8 + 10752u_i^{10} + 135168u_i^{12} + \cdots \\ &= u_i^2(1 + 8u_i^2)(1 + 80u_i^4)(1 + 256u_i^6)(1 + 8704u_i^8) \cdots \end{aligned}$$

$X_0(16)$:

$$\begin{aligned} v_{i+1} &= v_i^2 + 4v_i^6 + 32v_i^{10} + 320v_i^{14} + 3584v_i^{18} + 43008t_i^{22} + \cdots \\ &= v_i^2(1 + 4v_i^4)(1 + 32v_i^8)(1 + 192v_i^{12})(1 + 2816v_i^{16})(1 + 25600v_i^{20}) \cdots \end{aligned}$$

Similarly the first few classes of algorithms on $X_0(3^n)$ give rise to the following p -adic analytic recursions.

$X_0(3)$:

$$\begin{aligned} s_{i+1} &= s_i^3 - 36s_i^4 + 1026s_i^5 - 27216s_i^6 + 702027s_i^7 - 17898408s_i^8 + \cdots \\ &= s_i^3(1 - 36s_i)(1 + 1026s_i^2)(1 + 9720s_i^3) \cdots \\ &= s_i^3(1 - 36s_i)(1 + 1026s_i^2)(1 + 9720s_i^3) \cdots \\ &\quad (1 + 1051947s_i^4)(1 + 9998964s_i^5 + 93927276s_i^6) \cdots \end{aligned}$$

$X_0(9)$:

$$\begin{aligned} t_{i+1} &= t_i^3 - 9t_i^4 + 54t_i^5 - 252t_i^6 + 891t_i^7 - 1701t_i^8 - 6426t_i^9 + \cdots \\ &= t_i^3(1 - 9t_i - 252t_i^3)(1 + 54t_i^2 + 649674t_i^6)(1 + 5265t_i^4) \cdots \\ &\quad (1 + 486t_i^3 + 33048t_i^5 + 2925234t_i^7 + 98492517t_i^9) \cdots \end{aligned}$$

The above power series give explicit convergent series for the action of the Frobenius automorphism on ordinary CM points on the modular curves $X_0(p^n)$ for those particular values of p and n . The power product representations have the property that all but finitely many terms equal one to any fixed precision p^i . Note that the iteration $x_i \mapsto x_{i+1}$ is of the form $x_{i+1} = x_i^p f(x_i)$ for some power series $f(x_i)$ in x_i , and that the p -th powering gains relative precision. Thus in the initial phase we iterate the initial terms of the power product representation mod p^i to lift an approximation to the CM point as described in Table 1.

Table 2. Modular Action of Verschiebung.

m -th power of Verschiebung	Norm-equivalent expression	m
$X_0(2) :$		
$\frac{(256s_2 + 1)(-256s_2(256s_2 + 1) + 16s_1 + 1)}{(256s_1 + 1)(512s_2(64s_2 + 1) - 8s_1 + 1)}$	$\frac{(-256s_2(256s_2 + 1) + 16s_1 + 1)}{(512s_2(64s_2 + 1) - 8s_1 + 1)}$	2
$X_0(4) :$		
$\frac{32t_2 + 1}{8t_1 + 1}$	$\frac{32t_1 + 1}{8t_1 + 1}$	2
$X_0(8) :$		
$\frac{(-4u_1 + 1)(4u_2 + 1)}{4u_2 - 1}$	$1 + 4u_1$	1
$X_0(16) :$		
$\frac{(-4v_1^2 + 1)(4v_2^2 + 1)}{4v_2^2 - 1}$	$1 + 4v_1^2$	1
$X_0(3) :$		
$\frac{(3s_1 + 1)(-19683s_1^2 - 486s_1 + 1)}{(243s_1 + 1)(-27s_1^2 + 18s_1 + 1)}$		2
$X_0(9) :$		
$\frac{(3t_1 + 1)(27t_1^2 + 1)(-243(81(27t_1^2 + 9t_1 + 1)^2t_1^2 + 2(27t_1^2 + 9t_1 + 1)t_1 + 1))}{(-27t_1^2 + 1)(243(27t_1^2 + 9t_1 + 1)t_1 + 1)(729t_1^4 + 486t_1^3 + 162t_1^2 + 18t_1 + 1)}$		2

2.3 Action of Verschiebung

In order to apply the Heegner point constructions to the determination of the trace of Frobenius, we need to pullback of Frobenius between the differentials of parametrized curves specified by a modular correspondence. In Table 2 below we give the value of this scalar action of Verschiebung, the dual to Frobenius, in the left hand column. Using the identity $N(x_1) = N(x_2)$ for any Galois conjugates x_1 and x_2 , we are able to simplify the expressions by eliminating terms whose norm reduces to 1. In the final column we indicate with a 1 or 2 whether the expression is for the Verschiebung itself, or its square. In the latter case, we must extract a square root in the course of computing the norm.

3 Algorithm and Performance

In order to construct the initial lifting of a finite field element to a p -adic element with precision w , we make use of the power series for x_{i+1} in terms of x_i as described in Table 1. Since the power series is approximated mod p by the congruence $x_{i+1} \equiv x_i^p \pmod{p}$, each application of this p -adic analytic function gains one coefficient of precision.

The analytic method, using a precomputed power product representation of the Hensel lifting of the power series appears to be more efficient than a naive linear Hensel lifting to compute the canonical lift to a precision of one 32-bit

Table 3. HeegnerPointBlockLift.

Input: The modular polynomial Φ , integers m and w , and a p -adic element x such that (x, x^σ) is a zero of Φ to precision p^w .

Output: A lift of x such that (x, x^σ) is a zero to precision p^{mw} .

```

 $D_X = \Phi_X(x, x^\sigma) \bmod p^w$ 
 $D_Y = \Phi_Y(x, x^\sigma) \bmod p^w$ 
for  $(1 \leq i \leq m)$  {
   $R_x = (\Phi(x, x^\sigma) \operatorname{div} p^{iw}) \bmod p^w$ 
  for  $(1 \leq j \leq w)$  {
     $\Delta_X = (R_x \bmod p)^{1/p}$  lifted to precision  $p^w$ 
     $R_x = (R_x + D_X \Delta_X + D_Y \Delta_X^\sigma)/p$ 
     $x \mathrel{+}= p^{iw+j} \Delta_X$ 
  }
}
return  $x$ 

```

computer word. This is in part explained by the observation that a significant number of steps of the $f_i(x)$'s are in fact equal to 1, and so can be omitted from the product. Finally, we note that this product expression structures the Hensel lifting to use only multiplications.

The second phase of the lifting mirrors Algorithm 1 of SST [11], expressed here in terms of the Frobenius automorphism σ rather than its inverse. The algorithm of SST refers to the classical modular polynomial $\Phi_p(j_1, j_2)$ relating the j -invariants of two p -isogenous curves, but in fact applies in great generality² to find p -adic solutions to a bivariate polynomial $\Phi(x, y)$ for which $(x^p - y) \mid \Phi(x, y) \bmod p$.

Here we apply it to our modular correspondences $\Phi(x, y)$ on the curves $X_0(N)$. We define $\Phi_X(x, y)$ and $\Phi_Y(x, y)$ be the derivatives with respect to the first and second variable, respectively, of the modular correspondence. The algorithm is given in Table 3.

The final step is to make use of the precomputed form of the action Frobenius on the differentials for an elliptic curve parametrization by $X_0(N)$. This action will be a rational function $\pi_1 = \pi(x_1)$ in the value x_1 of the lifted point. The Frobenius endomorphism is the product of the Galois conjugate Frobenius isogenies, so the norm $N(\pi_1)$ of this value gives the action of the Frobenius endomorphism on the differentials. Since the minimal polynomial $X^2 - tX + q$ for this element is congruent to $X(X - t)$ modulo q , we see that $N(\pi_1) \bmod q \equiv 0$ and $(q \operatorname{div} N(\pi_1)) \equiv t \bmod q$. In a now standard trick, the norm is computed using the identity $N(\pi_1) = \exp(\operatorname{Tr}(\log(\pi_1)))$, using the efficiency of trace computation [11].

An generic implementation [7] of the method in Magma [8] yields the following timing data of Table 4 on an 1.4GHz AMD machine. The algorithm

² This observation was already used by Gaudry [5] in extending this algorithm to a modified AGM modular equation.

Table 4. Timing Data for AGM- $X_0(N)$.

p = 2:	m	$\log_2(q)$	$X_0(2)$	$X_0(4)$	$X_0(8)$	$X_0(16)$
	163	163.00	0.48s	0.46s	0.45s	0.55s
	193	193.00	0.61s	0.59s	0.60s	0.72s
	239	239.00	0.91s	0.88s	0.91s	1.08s
p = 3:			$X_0(3)$	$X_0(9)$		
	103	163.25	8.95s	10.8s		
	121	191.78	19.7s	19.8s		
	127	201.29	21.1s	21.2s		
	151	239.33	43.5s	46.6s		
p = 5:			$X_0(5)$	$X_0(25)$		
	71	164.86	8.06s	8.75s		
	83	192.72	12.6s	13.5s		
	103	239.16	30.5s	30.9s		
p = 7:			$X_0(7)$			
	59	165.63	5.13s			
	69	193.70	10.9s			
	71	199.32	11.3s			
	83	233.01	19.8s			
	85	238.63	21.6s			
p = 13:			$X_0(13)$			
	43	159.12	4.18s			
	53	196.12	8.66s			
	61	225.73	14.3s			
	65	240.53	19.1s			

makes use of the internal Magma implementation of an efficient Galois action on unramified cyclotomic extensions when $p = 2$, and otherwise falls back on Hensel lifting to determine Galois images when the residue characteristic is odd. The timings listed are independent of the one-time setup costs for initializing the p -adic lifting rings. Further specific optimizations for $p = 2$ make this case comparatively faster than for odd residue characteristic.

4 Relations with Other Algorithms

The chosen model curve for $X_0(8)$ is the equation $u_1^2(4u_2 + 1)^2 = u_2$, which has the property that its reduction modulo 2 takes the form $u_1^2 = u_2$, so that u_2 is the Galois image of u_1 . Over a field of characteristic zero, this equation becomes isomorphic to the equation arising in the “univariate” version of the AGM recursion $4xy^2 = (x + 1)^2$ via the change of variables³

$$x = \frac{1 + 4u_1}{1 - 4u_1} \text{ and } y = \frac{1 + 4u_2}{1 - 4u_2}.$$

³ Gaudry [5] makes a similar change of variables $x = 1/(1 + 8u)$ and $y = 1/(1 + 8v)$, from which he obtains the relation $(u + 2v + 8uv)^2 + (4u + 1)v$, having the similar property of giving rise to an equation $u^2 = v$ between Galois conjugates modulo 2.

Thus the use of 2-adic Heegner point lifts on $X_0(8)$ to determine the number of points on an elliptic curve over \mathbb{F}_{2^m} could fall under a purported patent application on the AGM point counting method⁴.

In contrast, the modular curves $X_0(1)$, $X_0(2)$, $X_0(4)$, $X_0(8)$, or $X_0(16)$ are nonisomorphic as moduli spaces, and only the modular correspondence for $X_0(8)$ transforms by change of variables into the univariate AGM method. In fact if j is a root of the polynomial $x^3 + x + 1$ in \mathbb{F}_2 , then the canonical lift of j on $X_0(1)$ is a root of the polynomial:

$$x^3 + 3491750x^2 - 5151296875x + 12771880859375.$$

The original method of Satoh, extended to characteristic 2 as in [4] or [11] finds some 2-adic approximation to a root of this polynomial. In contrast, in terms of the functions s , t , u , and v , the minimal polynomials over \mathbb{Q} of a canonical lift are respectively:

$$\begin{aligned} 2^{36}x^6 + 2^{25}83x^5 + 14351421440x^4 + 412493295x^3 + 3503765x^2 + 166x + 1, \\ 2^{24}x^6 + 2^{17}59x^5 + 1561856x^4 + 143007x^3 + 6101x^2 + 118x + 1, \\ 2^6x^6 + 2^{41}7x^5 + 572x^4 + 203x^3 + 13x^2 + 2x + 1, \\ 2^3x^6 - 4x^5 + 18x^4 + 13x^3 + 9x^2 + 4x + 1. \end{aligned}$$

The above polynomials are examples of class invariants obtained by modular correspondences on $X_0(1)$, $X_0(2)$, $X_0(4)$, $X_0(8)$, and $X_0(16)$, the latter examples naturally generalizing the construction of Couveignes and Henocq [2] for $X_0(1)$.

References

1. D. V. Bailey, C. Paar, Efficient arithmetic in finite field extensions with application in elliptic curve cryptography. *J. Cryptology*, **14** (2001), no. 3, 153–176.
2. J.-M. Couveignes and T. Henocq. Action of modular correspondences around CM points, *Algorithmic Number Theory (ANTS V, Sydney)*, 234–243, Lecture Notes in Computer Science, **2369**, Springer, Berlin, 2002.
3. N. Elkies. Elliptic and modular curves over finite fields and related computational issues, *Computational perspectives on number theory (Chicago, IL, 1995)*, 21–76, AMS/IP Stud. Adv. Math., **7**, Amer. Math. Soc., Providence, RI, 1998.
4. M. Fouquet, P. Gaudry, and R. Harley. An extension of Satoh’s algorithm and its implementation, *J. Ramanujan Math. Soc.*, **15** (2000), 281–318.
5. P. Gaudry, A comparison and a combination of SST and AGM algorithms for counting points of elliptic curves in characteristic 2, *Advances in Cryptology – ASIACRYPT 2002*, 311–327, Lecture Notes in Computer Science, **2501**, Springer, Berlin, 2002.
6. A. Kato, T. Kobayashi, and T. Saito. Use of the Odd Characteristic Extension Field in the Internet X.509 Public Key Infrastructure, PKIX Working Group, Internet Draft, <http://www.ietf.org/internet-drafts/draft-kato-pkix-ecc-oef-00.txt>

⁴ Note however that the U.S. patent application concerns the “non-converging AGM iteration” (refer to <http://argote.ch>), as in Mestre’s original binary AGM recursion [9], as distinct from Satoh’s prior algorithm, the subsequent published univariate AGM recursions, and the variants described herein.

7. D. Kohel, <http://magma.maths.usyd.edu.au/~kohel/magma>, 2003.
8. Magma Handbook, J. Cannon and W. Bosma, eds., <http://magma.maths.usyd.edu.au/magma/htmlhelp/MAGMA.htm>, 2003
9. J.-F. Mestre, Lettre à Gaudry et Harley. <http://www.math.jussieu/~mestre>, 2001.
10. T. Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *J. Ramanujan Math. Soc.* **15** (2000), no. 4, 247–270.
11. T. Satoh, B. Skjærnaa, and Y. Taguchi. Fast computation of canonical lifts of elliptic curves and its application to point counting, *Finite Fields Appl.* **9** (2003), no. 1, 89–101.
12. T. Satoh. On p -adic point counting algorithms for elliptic curves over finite fields, *Algorithmic number theory (ANTS V, Sydney)*, 43–66, Lecture Notes in Computer Science, **2369**, Springer, Berlin, 2002,
13. R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p . *Math. Comput.* **44** (1985) 483–494.

5 Appendix of Equations of Higher Level

In this appendix we give the equations for the modular correspondences and action of Verschiebung necessary to implement the AGM- $X_0(N)$ for $N = 5, 25, 7$, and 13 . The modular correspondences on $X_0(5)$, $X_0(25)$, $X_0(7)$, and $X_0(13)$ with respect to a degree one function on the curve are as follows.

$X_0(5)$:

$$\begin{aligned} s_1^5 &- 244140625s_1^4s_2^5 - 58593750s_1^4s_2^4 - 4921875s_1^4s_2^3 - 162500s_1^4s_2^2 \\ &- 1575s_1^4s_2 - 1953125s_1^3s_2^4 - 468750s_1^3s_2^3 - 39375s_1^3s_2^2 - 1300s_1^3s_2 \\ &- 15625s_1^2s_2^3 - 3750s_1^2s_2^2 - 315s_1^2s_2 - 125s_1s_2^2 - 30s_1s_2 - s_2 = 0 \end{aligned}$$

$X_0(25)$:

$$\begin{aligned} t_1^5 &- 625t_1^4t_2^5 - 625t_1^4t_2^4 - 375t_1^4t_2^3 - 125t_1^4t_2^2 - 25t_1^4t_2 - 625t_1^3t_2^5 - 625t_1^3t_2^4 \\ &- 375t_1^3t_2^3 - 125t_1^3t_2^2 - 25t_1^3t_2 - 375t_1^2t_2^5 - 375t_1^2t_2^4 - 225t_1^2t_2^3 - 75t_1^2t_2^2 \\ &- 15t_1^2t_2 - 125t_1t_2^5 - 125t_1t_2^4 - 75t_1t_2^3 - 25t_1t_2^2 - 5t_1t_2 - 25t_2^5 - 25t_2^4 \\ &- 15t_2^3 - 5t_2^2 - t_2 = 0 \end{aligned}$$

The functions s on $X_0(5)$ and t on $X_0(25)$ are linked by the relation

$$s = 25t^5 + 25t^4 + 15t^3 + 5t^2 + t.$$

$X_0(7)$:

$$\begin{aligned} s_1^7 &- 13841287201s_1^6s_2^7 - 7909306972s_1^6s_2^6 - 1856265922s_1^6s_2^5 - 224003696s_1^6s_2^4 \\ &- 14201915s_1^6s_2^3 - 422576s_1^6s_2^2 - 4018s_1^6s_2 - 282475249s_1^5s_2^6 - 161414428s_1^5s_2^5 \\ &- 37882978s_1^5s_2^4 - 4571504s_1^5s_2^3 - 289835s_1^5s_2^2 - 8624s_1^5s_2 - 5764801s_1^4s_2^5 \\ &- 3294172s_1^4s_2^4 - 773122s_1^4s_2^3 - 93296s_1^4s_2^2 - 5915s_1^4s_2 - 117649s_1^3s_2^4 \\ &- 67228s_1^3s_2^3 - 15778s_1^3s_2^2 - 1904s_1^3s_2 - 2401s_1^2s_2^3 - 1372s_1^2s_2^2 - 322s_1^2s_2 \\ &- 49s_1s_2^2 - 28s_1s_2 - s_2 = 0 \end{aligned}$$

$X_0(13)$:

$$\begin{aligned}
& s_1^{13} - 23298085122481s_1^{12}s_2^{13} - 46596170244962s_1^{12}s_2^{12} - 44804009850925s_1^{12}s_2^{11} \\
& - 27020264402404s_1^{12}s_2^{10} - 11283187332872s_1^{12}s_2^9 - 3409754413780s_1^{12}s_2^8 \\
& - 758378576462s_1^{12}s_2^7 - 123855918940s_1^{12}s_2^6 - 14548002326s_1^{12}s_2^5 \\
& - 1174999540s_1^{12}s_2^4 - 59916584s_1^{12}s_2^3 - 1623076s_1^{12}s_2^2 - 15145s_1^{12}s_2 \\
& - 1792160394037s_1^{11}s_2^{12} - 3584320788074s_1^{11}s_2^{11} - 3446462296225s_1^{11}s_2^{10} \\
& - 2078481877108s_1^{11}s_2^9 - 867937487144s_1^{11}s_2^8 - 262288801060s_1^{11}s_2^7 \\
& - 58336813574s_1^{11}s_2^6 - 9527378380s_1^{11}s_2^5 - 1119077102s_1^{11}s_2^4 \\
& - 90384580s_1^{11}s_2^3 - 4608968s_1^{11}s_2^2 - 124852s_1^{11}s_2 - 137858491849s_1^{10}s_2^{10} \\
& - 275716983698s_1^{10}s_2^{10} - 265112484325s_1^{10}s_2^9 - 159883221316s_1^{10}s_2^8 \\
& - 66764422088s_1^{10}s_2^7 - 20176061620s_1^{10}s_2^6 - 4487447198s_1^{10}s_2^5 \\
& - 732875260s_1^{10}s_2^4 - 86082854s_1^{10}s_2^3 - 6952660s_1^{10}s_2^2 - 354536s_1^{10}s_2 \\
& - 10604499373s_1^9s_2^{10} - 21208998746s_1^9s_2^9 - 20393268025s_1^9s_2^8 \\
& - 12298709332s_1^9s_2^7 - 5135724776s_1^9s_2^6 - 1552004740s_1^9s_2^5 \\
& - 345188246s_1^9s_2^4 - 56375020s_1^9s_2^3 - 6621758s_1^9s_2^2 - 534820s_1^9s_2 \\
& - 815730721s_1^8s_2^9 - 1631461442s_1^8s_2^8 - 1568712925s_1^8s_2^7 - 946054564s_1^8s_2^6 \\
& - 395055752s_1^8s_2^5 - 119384980s_1^8s_2^4 - 26552942s_1^8s_2^3 - 4336540s_1^8s_2^2 \\
& - 509366s_1^8s_2 - 62748517s_1^7s_2^8 - 125497034s_1^7s_2^7 - 120670225s_1^7s_2^6 \\
& - 72773428s_1^7s_2^5 - 30388904s_1^7s_2^4 - 9183460s_1^7s_2^3 - 2042534s_1^7s_2^2 \\
& - 333580s_1^7s_2 - 4826809s_1^6s_2^7 - 9653618s_1^6s_2^6 - 9282325s_1^6s_2^5 - 5597956s_1^6s_2^4 \\
& - 2337608s_1^6s_2^3 - 706420s_1^6s_2^2 - 157118s_1^6s_2 - 371293s_1^5s_2^6 - 742586s_1^5s_2^5 \\
& - 714025s_1^4s_2^4 - 430612s_1^5s_2^3 - 179816s_1^5s_2^2 - 54340s_1^5s_2 - 28561s_1^4s_2^5 \\
& - 57122s_1^4s_2^4 - 54925s_1^4s_2^3 - 33124s_1^4s_2^2 - 13832s_1^4s_2 - 2197s_1^3s_2^4 \\
& - 4394s_1^3s_2^3 - 4225s_1^3s_2^2 - 2548s_1^3s_2 - 169s_1^2s_2^3 - 338s_1^2s_2^2 \\
& - 325s_1^2s_2 - 13s_1s_2^2 - 26s_1s_2 - s_2 = 0
\end{aligned}$$

We note that a canonical lift only exists for the invariants of ordinary curves. The supersingular points, in contrast, fail to converge, and are in fact poles of each the chosen functions for the lifting process. In characteristics 2 and 3 the j -invariant 0 is supersingular, which explains why we take as starting point of our canonical lifting algorithm $1/j \equiv s_1 \equiv t_1 \cdots \pmod{p}$. For p equal to 5 the j -invariant of a supersingular curve is also 0, and the starting point of lifting is therefore also $1/j \equiv s_1 \equiv t_1 \pmod{5}$. However for 7 and 13 the starting points of the lifting algorithms are $1/(j+1) \equiv s_1 \pmod{7}$ and $1/(j-5) \equiv s_1 \pmod{13}$, corresponding to the supersingular j -invariants 6 and 5, respectively.

To complete the specification of the algorithms for $X_0(5)$, $X_0(7)$, and $X_0(13)$, it remains to give the action of Verschiebung on the differentials of a generic curve as in Table 2. In terms of a special value s_1 which is the canonical lift of the invariants of an ordinary elliptic curve, we find the following form for square of the action of pullback by the Verschiebung on two parametrized curves.

$$\text{X}_0(5): \quad -\frac{G_5(s_1, 1)H_5(5^3s_1, 1)}{G_5(5^2s_1, 1)H_5(1, s_1)}, \text{ where } \begin{cases} G_5(X, Y) = 5X^2 + 10XY + Y^2, \\ H_5(X, Y) = -X^2 - 4XY + Y^2. \end{cases}$$

$$\text{X}_0(7): \quad -\frac{F_7(s_1, 1)(-7^7s_1^4 + G_7(7^2s_1, 1) + 1)}{F_7(7^2s_1, 1)(-7s_1^4 + 7G_7(1, s_1) + 1)}, \text{ where }$$

$$F_7(X, Y) = X^2 + 5XY + Y^2,$$

$$G_7(X, Y) = (2X^2 + 9XY + 10Y^2)XY.$$

$$\underline{X_0(13)}: -\frac{G_{13}(1, s_1)H_{13}(13s_1, 1)}{G_{13}(13s_1, 1)H_{13}(1, s_1)}, \text{ where}$$

$$G_{13}(X, Y) = X^4 + 7X^3Y + 20X^2Y^2 + 19XY^3 + Y^4,$$

$$H_{13}(X, Y) = X^6 + 10X^5Y + 46X^4Y^2 + 108X^3Y^3 + 122X^2Y^4 + 38XY^5 - Y^6.$$

The action of Frobenius with respect to t_1 on $X_0(25)$ is determined by means of the expression for the function $s_1 = 25t_1^5 + 25t_1^4 + 15t_1^3 + 5t_1^2 + t_1$ on $X_0(5)$ in terms of the function t_1 on $X_0(25)$.

Key Management Schemes for Stateless Receivers Based on Time Varying Heterogeneous Logical Key Hierarchy

Miodrag J. Mihaljević

Mathematical Institute
Serbian Academy of Sciences and Arts
Kneza Mihaila 35, 11001 Belgrade, Serbia and Montenegro
`miodragm@turing.mi.sanu.ac.yu`

Abstract. This paper proposes a family of key management schemes for broadcast encryption based on a novel underlying structure - Time Varying Heterogeneous Logical Key Hierarchy (TVH-LKH). Note that the main characteristics of the previously reported key management schemes include the following: employment of a static underlying structure for key management, and addressing the subset covering problem over the entire underlying structure. Oppositely, the main underlying ideas for developing of the novel key management schemes based on TVH-LKH include the following: (i) employment of a reconfigurable underlying structure; and (ii) employment of a divide-and-conquer approach related to the underlying structure and an appropriate communications-storage-processing trade-off (for example, a small increase of the communication overload and large reduction of the storage and processing overload) for addressing the subset covering problem and optimization of the overloads. The design is based on a set of “static” keys at a receiver (stateless receiver) which are used in all possible reconfiguration of the underlying structure for key management, and accordingly, in a general case, a key plays different roles depending on the employed underlying structure. A particular family of the components for developing TVH-LKH, is also proposed and discussed. The proposed technique is compared with the recently reported schemes, and the advantages of the novel one are pointed out.

Keywords: broadcast encryption, stateless receivers, key management, time varying schemes, heterogeneous structures, reconfigurability, tree graphs.

1 Introduction

Broadcasting encryption (BE) schemes define methods for encrypting content so that only privileged users are able to recover the content from the broadcast. Later on, this flagship BE application has been extended to another one - media content protection (see [17] or [12], for example). This application has the same one-way nature as an encrypted broadcast: A recorder makes an encrypted

recording and, a player needs to play it back. This situation usually does not allow opportunity for the player and recorder to communicate. Accordingly, in this paper we are dealing with the stateless receivers - the devices in which the operations must be accomplished based only on the current transmission and its initial configuration because these receivers do not have a possibility to update their state from session to session.

When cryptography is used for securing communications, a session- encrypting key (SEK) is used to encrypt the data. Since the data are distributed to multiple receivers, in order to reduce the amount of encryption at the sender node and to minimize the required bandwidth, every intended receiver as well as the sender should share an identical SEK. In order to ensure that only the valid members of the group have access to the communications, SEK needs to be changed whenever the lifetime of it expires, or there is a change in membership of the group, or one or more members are compromised. SEK needs to be updated under membership change for the following reasons: (i) when a new member joins, to ensure that the new member has no access to the past communication of the group, and (ii) when a member departs or is deleted, to ensure that the departed or deleted member does not have access to future communications.

Ensuring that only the valid members of the selected group have SEK at any given time instance is the key management problem in BE. On the other hand, for the SEK updating, a system needs another set of keys called the key-encrypting keys (KEKs) that can be used to encrypt and transmit the updated SEK to the valid members of the group. Hence, the key management problem reduces to the problem of distributing the KEKs to the members such that at any given time instant all the valid members can be securely reached and updated with the new SEK.

A number of sophisticated methods for BE key management have been reported in the literature employing the following approach: Provide in advance the receivers with a collection of the keys (KEKs) in such a manner that the communication overload is reduced.

The first breakthrough in BE key management is reported in [8] where the schemes in which each receiver has a fixed set of reusable keys were proposed. However, the complexity of these schemes was strongly dependent on the size of the adversarial coalition.

Later on, a number of different schemes as well as the system approaches, have been reported and analyzed - see [16], [20]-[21], [3], [1], [9], [17], [18], [19], [2] and [4], for example, and recently, certain results have been reported in [11], [13], [6], [5], [14] and [15], as well.

According to [11], the most interesting variant of BE deals with stateless receivers and has the following requirements:

- Each user is initially given a collection of symmetric encryption keys.
- The keys can be used to access any number of broadcasts.
- The keys can be used to define any subset of users as privileged.
- The keys are not affected by the user's "viewing history".
- The keys do not change when other users join or leave the system.

- Consecutive broadcasts can address unrelated privileged subsets.
- Each privileged user can decrypt the broadcast by himself.
- Even a coalition of all non-privileged users cannot decrypt the broadcast.

This paper addresses the problem of developing improved BE key management schemes assuming the above given requirements.

Contributions of the paper

This paper proposes a family of key management schemes for broadcast encryption based on the Time Varying Heterogeneous Logical Key Hierarchy (TVH-LKH).

Note that the main characteristics of the previously reported key management schemes include the following ones: (i) employment of a static underlying structure for key management; (ii) addressing the subset covering problem considering the underlying structure as a whole.

Oppositely, the main underlying ideas for developing of the improved key management schemes based on TVH-LKH include the following:

- employment of a time varying (reconfigurable) heterogeneous underlying structure;
- employment of a divide-and-conquer approach related to the underlying structure and an appropriate communications-storage-processing trade-off (for example: a small increase of the communication overload and large reduction of the storage and processing overload) for addressing the subset covering problem and optimization of the overloads.

Note that the proposed design is based on a set of “static” keys at a receiver (stateless receivers) which are used for all possible reconfiguration of the underlying structure for key management. So, in a general case, a key plays different roles depending on the employed underlying structure.

A family of the components called sectioned heterogeneous LKH (SH-LKH) and its special form consisting of the sectioned key trees (SKTs) are considered for developing the reconfigurable logical key hierarchy, and TVH-LKH with two particular family members called SKT-A and SKT-B is discussed.

The approach employed for design of SH-LKH family could be formulated as follows: Before dealing with the set covering issues, perform an appropriate preprocessing over the underlying LKH in order to specify a more suitable underlying structure for the set covering.

The main underlying ideas for developing a novel family of key management schemes are based on employment of appropriate clustering of the keys and users, and employment of the heterogeneous time varying and cluster oriented local key management. Accordingly, the design rationale for the novel family includes the following: (i) specification of the appropriate partitions/sections over the employed LKH; (ii) performing key management on the section-by-section basis; (iii) in a general case, employment different key management schemes in different sections or in different time instants; (iv) in certain cases, employment of modified local (section related) key management schemes which employ a relaxed specification of the privileged set.

Let N be the number of receivers and R the number of revocations. Assuming that the parameters of a particular TVH-LKH scheme with SKT-A and SKT-B are H_{0A} , H_{0B} , H_{1B} , R_{0A} , R_{0B} and R_{1B} , such that $1 \leq H_{0A} < \log_2 N$, $2 \leq H_{0B} + H_{1B} < \log_2 N$, $1 \leq R_{0A} \leq R$, and $1 \leq R_{1B} \leq R_{0B} \leq R$, its main characteristics are as follows. Dimension of the storage@receiver overload: $O(\max\{((H_{0A})^{1.5} - H_{0A} + \log_2 N), ((H_{0B})^{1.5} + (H_{1B})^{1.5} - H_{0B} - H_{1B} + \log_2 N)\})$. Dimension of the communications overload: $O(\min\{(R + R_{0A}((\log_2 N) - H_{0A}) - R_{0A} \log_2 R_{0A}), (R + R_{0B} + R_{1B}((\log_2 N) - H_{1B} - H_{0B}) - R_{1B} \log_2 R_{1B})\})$. Maximum dimension of the processing@receiver overload: $O(\max\{H_{0A}, \max\{H_{0B}, H_{1B}\}\})$.

An illustrative comparison of the main characteristics of the proposed key management and the recently reported ones is given in Table 1, assuming a huge group with a heavy dynamics in order to demonstrate advantages of the proposal even in the considered scenario. Intentionally, the comparison is related to the most powerful recently reported schemes based on the binary tree approach to demonstrate advantages of the considered particular TVH-LKH which is also based on the binary tree approach.

Table 1. Illustrative numerical comparison of the main characteristics of the proposed TVH-LKH key management schemes and the Complete Sub-Tree (CST) [17], Subset Difference (SD) [17] and Layered Subset Difference (LSD) [11], assuming $N = 2^{27}$ receivers and $R = 2^{15}$ revocations, and that the parameters of the considered TVH-LKH technique are $H_{0A} = 10$, $H_{0B} = 7$, $H_{1B} = 7$, $R_{0A} = 2^{14}$, $R_{0B} = 2^{14}$ and $R_{1B} = 2^{11}$.

technique	storage@receiver	processing@receiver	communication
CST [17]	~ 27	~ 5	$\sim 12 \cdot 2^{15}$
SD [17]	~ 729	~ 27	$\sim 2^{15}$
basic LSD [11]	~ 140	~ 27	$\sim 2^{15}$
proposed TVH-LKH	~ 49	~ 7	$\sim 1.5 \cdot 2^{15}$

Table 1 illustrates how combining of the heterogeneous schemes in a time-varying manner appear as a powerful approach for developing improved key management schemes which yield a possibility for appropriate trade-offs between the main overloads of the system.

Organization of the paper

Section 2 yields the underlying ideas for developing of the improved key management schemes, and a general framework for key management based on the reconfigurable logical key hierarchy (TVH-LKH). Key management based on reconfigurable logical key hierarchy which employs a collection of the sectioned

key trees is considered in Section 3 including a comparison of a particular TVH-LKH based technique and recently reported schemes targeting the same key management scenario. Finally, some concluding discussions are given in Section 4, and two proposition proofs are accommodated in Appendices A-B.

2 Underlying Ideas and General Framework for a Novel Design

This section points out the underlying ideas for the improved key management schemes proposed in this paper, and a general framework for development of these schemes.

Note that the *general static key management paradigm* is based on the following:

- (a) BE center specify a set of all keys it will use, and assigns its subset to each receiver in such a manner that based on the keys stored at the receivers, BE center can split the set of all receivers into two arbitrary (usually) non overlapping parts.
- (b) BE center adopts a method for covering an arbitrary subset of the receivers taking into account the keys assigned to the receivers.
- (c) The established system is used for the session key distribution.

Unfortunately, in a general case, the above item (b) is a variation of the Set Cover problem (see [10] for example): It is known that no approximation algorithm exists for the Set Cover with a worst-case approximation ratio better than $\ln(N)$ [7] (assuming that N is the number of receivers).

In order to deal with the covering problem in an efficient way and employing much smaller required set of keys and the reduced processing at a receiver in comparison with the reported schemes, this section proposes a novel approach based on the reconfigurable key management. The following main three issues are addressed: (i) underlying ideas for proposing reconfigurable logical key hierarchy; (ii) general framework for the reconfigurable key management; and (iii) a discussion on selection of the main components for the proposed framework.

2.1 Underlying Ideas for the Key Management Schemes Based on Reconfigurable Logical Key Hierarchy

Recall that the main characteristics of the reported key management schemes include the following:

- employment of a static underlying structure for the key management;
- addressing the subset covering problem considering the underlying structure as a whole.

Oppositely, the main underlying ideas for developing the improved TVH-LKH based key management schemes include the following:

- employment of a reconfigurable underlying structure;
- employment of a divide-and-conquer approach related to the underlying structure and an appropriate communications-storage-processing trade-off (for example, a small increase of the communication overload and large reduction of the storage and processing needed by a receiver) for addressing the subset covering problem and optimization of the system overloads.

Note that the design is based on a set of “static” keys at a receiver which are used for all possible reconfiguration of the underlying structure for key management. So, in a general case, a particular key plays different roles depending on the employed underlying structure.

Recently, very efficient key management schemes Complete SubTree (CST) and Subset Difference (SD) have been proposed in [17] and Layered Subset Difference (LSD) has been reported in [11]. These schemes have been developed by focusing on obtaining a solution for the underlying set covering problem using the tree based paradigm. The approach proposed in this paper, beside employment of the reconfigurability concept, is also different in comparison with the previously reported ones in a way which could be formulated as follows: Before dealing with the set covering issues, perform an appropriate preprocessing over the underlying LKH in order to specify a more suitable underlying structure for the set covering. The employed preprocessing could also be considered as a particular divide-and-conquer method for key management.

The main underlying ideas for developing a novel family of the key management schemes include the following ones.

- employment of time varying logical key hierarchy;
- specification of a set of different and appropriate partitions/sections of the logical key hierarchy (in a particular case based on appropriate clustering of the keys and users);
- performing key management on the section-by-section basis (heterogeneous cluster oriented local key management);
- in a general case, employment different key management schemes in different sections or the time instances;
- optionally, in certain cases, employment of modified local (section related) key management schemes which provide a relaxed specification of the privileged set.

The opportunity for employment of different key management schemes in different sections or the time instances opens a door for desired optimization of the key management overload characteristics. For example, recall that CST re-keying requires significantly smaller storage@receiver overload at the expense of increased communications overload in comparison with LSD based re-keying. Accordingly, employing the CST based technique in one subset of the tree sections and LSD based one in another subset, for example, yields an opportunity for obtaining the desired overall characteristics. Also note the following two characteristics of SD and LSD schemes: (i) communications overload is linear with R ; (ii) storage@receiver overload is polynomial with $\log N$. These characteristics

open a door for the trade-off based on divide-and-conquer approach. Additionally, note that, for example, a relaxed version of SD or LSD, which does not perform the strict revocations but the relaxed ones in a manner similar to that reported in [1], could be employed as the appropriate one in certain cases.

Also note that, although the key management at the center's side is time varying and based on the section-by-section processing, this has no impact at the receivers side, and after all, a receiver should employ, in an appropriate manner, just one of its KEKs to recover the new SEK.

2.2 General Framework for the Key Management Based on Reconfigurable Logical Key Hierarchy

The Center's Framework

Pre-processing

Establishing the reconfigurable logical key hierarchy based key management requires the following main actions at the center side.

- Specification of a collection of the underlying structures to be used for the covering of privileged (non-revoked) receivers.
- Assigning a set of keys to each of the receivers in such a manner that the key management can be performed employing any of the underlying structures from the collection.

Processing

For delivering a new SEK the center performs the following:

- According to the given list of revocations, the center select an appropriate underlying structure from the collection for key management.
- The center jointly broadcast encrypted forms of the new SEK obtained by employing different KEKs and information of the KEKs employed, as well as the mode of their use, determined by currently selected underlying structure.

The Receiver's Framework

The framework for the proposed TVH-LKH based key management at the receiver's side consists of the following components:

- Each receiver is provided with a set of the keys and information on modes of their use.
- If not revoked, during the key management communication, a receiver obtains the following information:
 - which of its KEKs should be employed for the new SEK recovering, and
 - in which mode the employed KEK should be used (depending on the currently employed underlying structure from the predefined set),
 and accordingly it is able to recover the new SEK.

2.3 On the Keys Employment and Selection of the Underlying Structures

Note that the design is based on a set of “static” keys at a receiver which are used for all possible reconfiguration of the underlying structure for key management, and accordingly, in a general case, a key plays different roles depending on the employed underlying structure.

A main component of the reconfigurable key management is a collection of the underlying structures, and regarding these structures note the following.

- The underlying structures could be very different but all of them should fulfil the following condition: They should be able to work with the same single set of keys (KEKs) assuming that a key can be employed in different modes.
- A large number of the reconfigurable schemes can be designed in an ad-hock manner. Selection of the underlying structures included in the collection depends on the functional requirements of the key management. An optimized design should particularly take into account the space and time distribution of the revocations.

Accordingly, for given number of keys at a receiver, the reconfigurable logical key hierarchy (TVH-LKH) based key management yields an opportunity for minimizing the communications overload or the processing@receiver overload. On the other hand, note that TVH-LKH based schemes do not require additional storage@receiver overload in comparison with corresponding static LKH schemes which can be employed for the same revocation scenario.

3 A Reconfigurable Key Management Based on a Collection of Sectioned Heterogeneous LKHs

3.1 General Design Issues

Recall that the first step for establishing a reconfigurable logical key hierarchy is selection of a collection of the appropriate underlying structures for key management. This section proposes a particular TVH-LKH based on a novel structure called sectioned heterogeneous LKH (SH-LKH) for developing the underlying collection for the reconfigurable key management.

SH-LKH structure is displayed in Fig. 1. The triangles play roles of certain substructures: In a particular case they are the subtrees with the root at the triangle up and the leaves at the triangle bottom. These subtrees (embedded into triangles) could be very different including the following ones, (i) binary balanced tree, (ii) a tree consisting just of the root and a number of leaves, or (iii) other suitable trees.

From the center point of view, the key management scheme consists, as in an usual case, of the following two main components: (i) underlying structure for the keys and receivers assigning; (ii) methods employed for distributing a session key (SEK) to the stateless receivers. After this conceptual similarity, the proposed scheme differs from the reported ones as follows:

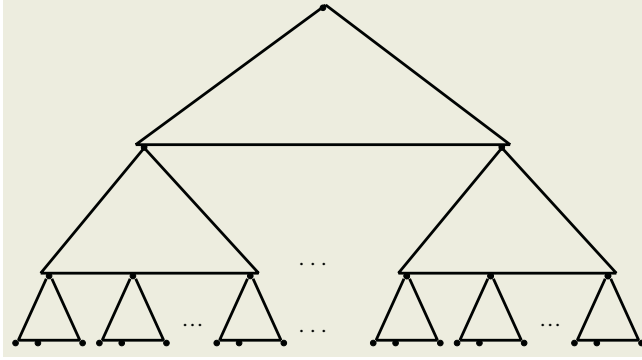


Fig. 1. A general form of the sectioned heterogeneous logical key hierarchy (SH-LKH). The triangles play roles of certain substructures, and in a particular case they are the subtrees with the root at the triangle up and the leaves at the triangle bottom.

- instead of a single underlying structure the center “possesses” a collection of different underlying structures
- each element of the collection is an SH-LKH;
- the distribution of SEK is based not on a single technique but on employment a number of different ones.

Accordingly, TVH-LKH employing SH-LKH is based on the following.

- The center selects an appropriate collection of SH-LKH to be used for key management.
- A set of keys is assigned to each of the receivers in such a manner that it can support any of SH-LKH key management schemes from the collection.
- In the case of SEK rekeying, the center broadcast SEK encrypted under different KEKs, and the related information on the employed keys and the mode of theirs use.
- At the receiver’s side the processing is adjusted according to the obtained information on the employed keys.

Note that a special case of SH-LKH is the sectioned key tree (SKT) introduced in [15].

3.2 Key Management Based on Sectioned Key Trees (SKTs)

This section, following [15], yields a background for developing and analyzing a particular TVH-LKH based on a collection of the underlying structures called sectioned key trees (SKTs). Note that SKTs are just a particular family of the binary tree structures which could be employed for design of certain TVH-LKH.

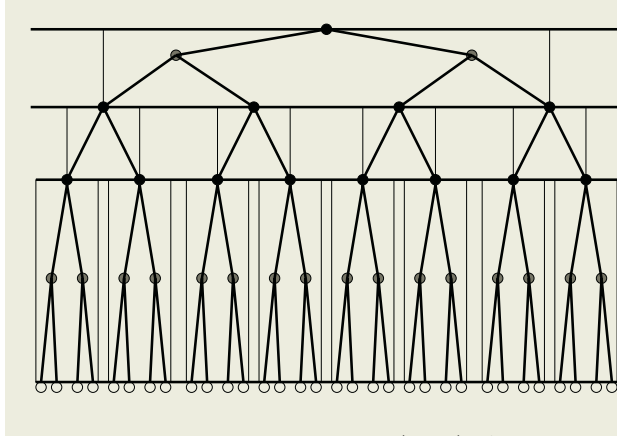


Fig. 2. An illustration of the sectioned key tree (SKT). As usually, the center is associated to the tree root, a receiver is at a leaf, and the keys are related to the tree nodes.

Family of SKTs

An SKT is the sectioned key tree displayed in Fig. 2 and obtained by the following horizontal and vertical partitioning:

- a number of the horizontal layers is specified;
- each layer is partitioned into a number of sections and each section contains a sub-tree which root is identical to a leaf of the upper layer section.

In a special case, the following can be enforced: each of the layers has the same height, and each layer's section contains the same number of nodes. Accordingly, each section contains the same subtree.

In a general case, the tree is partitioned into L horizontal layers with the heights H_ℓ , $\ell = 0, 1, \dots, L-1$, respectively, assuming that $\ell = 0$ corresponds to the bottom layer and $\ell = L-1$ to the top one. Then, the top layer contains a sub-tree with $2^{H_{L-1}}$ leaves, and a layer ℓ consists of

$$\prod_{i=\ell+1}^{L-1} 2^{H_i} = 2^{\sum_{i=\ell+1}^{L-1} H_i}$$

sections, each containing a sub-tree with 2^{H_ℓ} leaves.

Accordingly, we assume the following basic scenario for the key management based on the above underlying structure: N receivers grouped into M clusters, R revocations in total, assuming R_m revocations from a cluster with index m , $m = 1, 2, \dots, M$, and the parameter M is an integer such that $\sum_{m=1}^M R_m = R$ and N/M is an integer, $M \leq N$.

Section-by-Section Key Management

The proposed key management scheme assumes the section-by-section key management, and in a general case, it yields the opportunity for employment different local key management schemes in different sections.

Assuming SKT with L layers, and that a layer ℓ contains $M^{(\ell)}$ sections, $\ell = 0, 1, \dots, L - 1$, we propose the following section-by-section key management:

- layer 0 processing
 - For the subtree corresponding to section j , identify a set $\mathcal{R}_j^{(0)}$ of the leaves (receivers) which should be revoked, $j = 1, 2, \dots, M^{(0)}$.
 - Perform section-by-section processing: for the revocations over the subtree in section j employ a desired key management scheme for revocation of elements in $\mathcal{R}_j^{(0)}$, $j = 1, 2, \dots, M^{(0)}$.
- layer ℓ processing, $\ell = 1, 2, \dots, L - 1$
 - For the subtree corresponding to section j , identify a set $\mathcal{R}_j^{(\ell)}$ of the leaves which correspond to the sections in layer $\ell - 1$ affected by the revocations, and accordingly which should be revoked, $j = 1, 2, \dots, M^{(\ell)}$.
 - Perform section-by-section processing: for the revocations over the subtree in section j employ a desired key management scheme for revocation of elements in $\mathcal{R}_j^{(\ell)}$, $j = 1, 2, \dots, M^{(\ell)}$.

Center

At the center side, the procedure for revocation of a number of receivers consists of the following main steps:

- (a) the center specifies a set of receivers which should be revoked;
- (b) employing the section-by-section processing, the center decides on KEKs (nodes of the tree) which should be used for the new SEK delivery (encryption);
- (c) center broadcast the following message: (i) an implicit information on the employed KEKs; and (ii) the new SEK encrypted by each of the employed KEKs.

Let $E(\cdot)$ denotes the algorithm employed for encryption of the new SEK ($newSEK$), I_m defines the information on a KEK with index m , KEK_m , employed for encryption of the new SEK, $m = 1, 2, \dots, M$, where M is total number of KEKs employed for covering the desired subset of receivers, and $\mathcal{F}_{newSEK}(\cdot)$ denotes the algorithm employed for the payload encryption. Accordingly, BE center broadcast the following:

$$\begin{aligned}
 & [[I_1, I_2, \dots, I_M, E_{KEK_1}(newSEK), E_{KEK_2}(newSEK), \dots, \\
 & E_{KEK_M}(newSEK)], \mathcal{F}_{newSEK}(Payload)] \\
 = & [[I_1, I_2, \dots, I_M, C_1, C_2, \dots, C_M], PayloadCiphertext] .
 \end{aligned}$$

Receivers

At a receiver side the situation is equivalent as, for example, to the one when CST, SD, or LSD based approaches are employed. A receiver should store a number of cryptographic keys, monitor the communication channel to see whether

its current SEK should be exchanged, and if “yes” extract the new SEK based on certain processing employing a memorized key. Actually, a receiver can not be aware of the employed underlying structure at the center’s side.

At a receiver’s side the re-keying is performed as follows. Each receiver monitors the communications channel looking for the re-keying message broadcasted by the center. In this message, a non-revoked receiver will find an information on a KEK it posses which should be used for the new SEK recovering. Based on this information and the encrypted form of the new SEK, the non-revoked receiver will recover the new SEK.

Accordingly, upon receiving a broadcast message, the receiver performs the following operations:

- Finding I_m which is related to the receiver: If the receiver is revoked, no such information will be found;
- Employing I_m and the keys stored at the receiver, perform a processing in order to recover KEK_m employed for $newSEK$ encryption.
- Recovering the new SEK performing the decryption $E_{KEK_m}^{-1}(C_m)$.

Finally, after recovering the new SEK, the payload is obtained by

$$\mathcal{F}_{newSEK}^{-1}(PayloadCiphertext).$$

Two Particular Key Management Schemes: SKT-A and SKT-B

As the illustrative examples, this section specify two particular key management schemes called SKT-A and SKT-B where SKT stands for Sectioned Key Tree.

SKT-A. SKT-A is a particular key management scheme based on the following partitioning of the key tree and the local re-keying:

- There are two horizontal layers and height of the bottom one is equal to H_{0A} , and accordingly the upper layer has height equal to $\log_2 N - H_{0A}$;
- Basic LSD [11] revocation method is employed in each section of the bottom layer and CST [17] revocation method is employed in the upper layer-section.

SKT-B. SKT-B is a particular key management scheme based on the following partitioning of the key tree and the local re-keying:

- There are three horizontal layers and heights of the bottom and middle ones are equal to H_{0B} and H_{1B} , respectively; accordingly the top layer has height equal to $\log_2 N - H_{0B} - H_{1B}$;
- Basic LSD [11] revocation method is employed in each section of the two lower layers and CST [17] revocation method is employed in the upper layer-section.

Analysis of SKT Based Key Management Schemes

This section is focused on the following issues of the considered key management schemes: (i) communications - dimension of the messages overload to be sent for the re-keying; (ii) storage@receiver - dimension of keys which should be stored at a receiver; (iii) processing@receiver - processing overload due to the keys updating at receiver.

Main Characteristics of SKT-A. Taking into account the results reported in [17] and [11], it can be shown that SKT-A key management has the following main characteristics.

Proposition 1. SKT-A key management requires the following overload for R revocations in total which affect R_{0A} different sections, assuming R/R_{0A} revocations per section:

- dimension of the storage@receiver overload: $O((H_{0A})^{1.5} - H_{0A} + \log_2 N)$;
- dimension of the communications overload: $O(R + R_{0A}((\log_2 N) - H_{0A}) - R_{0A} \log_2 R_{0A})$;
- dimension of the processing@receiver overload: $O(H_{0A})$.

The proposition proof is given in Appendix A.

Main Characteristics of SKT-B. Taking into account the results reported in [17] and [11], it can be shown that SKT-B key management has the following main characteristics.

Proposition 2. SKT-B key management requires the following overload for R revocations in total which affect R_{0B} and R_{1B} different sections in the lower two layers, the bottom (0-th) and the middle (1-st) ones, respectively:

- dimension of the storage@receiver overload: $O((H_{0B})^{1.5} + (H_{1B})^{1.5} - H_{0B} - H_{1B} + \log_2 N)$;
- dimension of the communications overload: $O(R + R_{0B} + R_{1B}((\log_2 N) - H_{1B} - H_{0B}) - R_{1B} \log_2 R_{1B})$;
- dimension of the processing@receiver overload: $O(\max\{H_{0B}, H_{1B}\})$.

Proposition 2 proof is given in Appendix B.

3.3 Illustrative Example of TVH-LKH Employing SKTs

As an illustration of the proposed TVH-LKH based on a collection of SKTs, we consider the following toy example:

- TVH-LKH underlying collection consists of only SKT-A and SKT-B, and there are R revocations in total.
- In SKT-A case, R revocation affect R_{0A} clusters of receivers (sections).
- In SKT-B case, R revocation affect R_{0B} sections in the bottom layer and R_{1B} sections in the middle layer.

Proposition 3. The above specified TVH-LKH key management over a group of N receivers requires the following overload for R revocations in total which affect R_{0A} or R_{0B} and R_{1B} different sections in the lower layers, of SKT-A and SKT-B, respectively:

Table 2. Comparison of the storage@receiver and processing@receiver overloads of the proposed TVH-LKH key management scheme and the Complete Sub-Tree (CST)[17], Subset Difference (SD) [17] and Layered Subset Difference (LSD) [11], assuming N receivers, R revocations, and the parameters of the considered TVH-LKH technique are H_{0A} , H_{0B} , H_{1B} , R_{0A} , R_{0B} and R_{1B} , such that $1 \leq H_{0A} < \log_2 N$, $2 \leq H_{0B} + H_{1B} < \log_2 N$, $1 \leq R_{0A} \leq R$, and $1 \leq R_{1B} \leq R_{0B} \leq R$.

technique	storage@receiver	processing@receiver
CST [17]	$O(\log_2 N)$	$O(\log_2 \log_2 N)$
SD [17]	$O((\log_2 N)^2)$	$O(\log_2 N)$
basic LSD [11]	$O((\log_2 N)^{1.5})$	$O(\log_2 N)$
proposed TVH-LKH	$O(\max\{((H_{0A})^{1.5} - H_{0A} + \log_2 N), ((H_{0B})^{1.5} + (H_{1B})^{1.5} - H_{0B} - H_{1B} + \log_2 N)\})$	$O(H_{0A})$ or $O(\max\{H_{0B}, H_{1B}\})$

- dimension of the storage@receiver overload: $O(\max\{((H_{0A})^{1.5} - H_{0A} + \log_2 N), ((H_{0B})^{1.5} + (H_{1B})^{1.5} - H_{0B} - H_{1B} + \log_2 N)\})$;
- dimension of the communications overload: $O(\min\{(R + R_{0A}((\log_2 N) - H_{0A}) - R_{0A} \log_2 R_{0A}), (R + R_{0B} + R_{1B}((\log_2 N) - H_{1B} - H_{0B}) - R_{1B} \log_2 R_{1B})\})$;
- dimension of the processing@receiver overload: $O(H_{0A}, \text{ or } \max\{H_{0B}, H_{1B}\})$.

Proof Remarks. The proposition statement is a direct consequence of Propositions 1-2, and the selection strategy related to TVH-LKH which assumes employment of a scheme from the available collection which yields minimization of the communications overload. Particularly note that storage@receiver overload is determined by the maximum storage@receiver overload required by the schemes in the collection.

Accordingly, based on the results on CST, SD and LSD reported in [17] and [11], respectively, a comparison of these schemes and the considered TVH-LKH is summarized in Tables 2 and 3. Note that intentionally, the comparison is related to the most powerful recently reported schemes based on the binary tree approach to demonstrate advantages of considered particular TVH-LKH which is also based on the binary tree approach.

Table 2 yields a comparison of the storage and processing overloads, and Table 3 displays a comparison of the communications overloads.

4 Discussion

A novel and flexible paradigm for developing BE key management schemes is proposed. The proposal is based on the reconfigurability concept, and it yields

Table 3. Comparison of the communications overload of the proposed TVH-LKH key management scheme and the Complete Sub-Tree (CST)[17], Subset Difference (SD) [17] and Layered Subset Difference (LSD) [11], assuming N receivers, R revocations, and the parameters of the considered TVH-LKH technique are H_{0A} , H_{0B} , H_{1B} , R_{0A} , R_{0B} and R_{1B} , such that $1 \leq H_{0A} < \log_2 N$, $2 \leq H_{0B} + H_{1B} < \log_2 N$, $1 \leq R_{0A} \leq R$, and $1 \leq R_{1B} \leq R_{0B} \leq R$.

technique	communication overload
CST [17]	$O(R \log_2 \frac{N}{R})$
SD [17]	$O(R)$
basic LSD [11]	$O(R)$
proposed TVH-LKH	$O(\min\{ (R + R_{0A}(\log_2 N - H_{0A}) - R_{0A} \log_2 R_{0A}), (R + R_{0B} + R_{1B}((\log_2 N) - H_{1B} - H_{0B}) - R_{1B} \log_2 R_{1B}) \})$

the improved overall characteristics in comparison with the previously reported techniques. Tables 1-3 show that combining of the heterogeneous schemes in a time-varying manner appear as a powerful approach for developing improved key management schemes which yield a possibility for desired trade-offs between the main overloads related to the key management system. The design is based on a set of “static” keys at a receiver which are used for all possible reconfiguration of the underlying structure for key management, and accordingly, in a general case, a key plays different roles depending on the employed underlying structure.

The Gain Origins. The main origin of the gain obtained by the proposed key management in comparison with the previously reported techniques is due to the employed concept of reconfigurability and a dedicated divide-and-conquer approach. Particularly, certain gain origins include the following: (i) partition of the underlying LKH structure into the sections which appears as a very powerful technique for obtaining improved characteristics; (ii) performing overall key management based on a number of local (the section oriented) key managements; in a general case these key managements can be different and time varying.

Some Further Work Directions. TVH-LKH yields a generic framework for developing efficient key management, and besides the underlying structures discussed in this paper, it is an open problem to find novel constructions and particularly ones dedicated to certain applications. Also recall that (as in other schemes) there are three main overloads related to the proposed key management: storage@receiver, processing@receiver and communications overload. Taking into account certain constraints on these parameters, the proposed schemes can be optimized following the approaches reported in [3] and [19]. For example, for given constraints on storage@receiver and processing@receiver, the schemes can be optimized regarding the communications overload, or for the given communications

budget, the schemes can be optimized regarding storage@receiver and processing@receiver. On the other hand, in certain cases (where this is appropriate), further reduction of the overloads can be obtained employing a relaxed specification of the targeting receivers subset in a manner similar to that reported in [1] where certain receivers which should be revoked will not be excluded during the re-keying, assuming that the rate of this free-riders is within desired limits.

References

1. M. Abdalla, Y. Shavitt and A. Wool, "Key management for restricted multicast using broadcast encryption", *IEEE/ACM Trans. Networking*, vol. 8, pp. 443-454, Aug. 2000.
2. S. Banerjee and B. Bhattacharjee, "Scalable secure group communication over IP multicast", *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 1511-1527, Oct. 2002.
3. R. Canetti, T. Malkin and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption", *EUROCRYPT'99, Lecture Notes in Computer Science*, vol. 1592, pp. 459-474, 1999.
4. K.-C. Chan and S.-H. Gary Chan, "Distributed server networks for secure multicast", *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 1500-1510, Oct. 2002.
5. P. D'Arco and D.R. Stinson, "Fault tolerant and distributed broadcast encryption", *CT-RSA 2003, Lecture Notes in Computer Science*, vol. 2612, pp. 263-280, 2003.
6. G. Di Crescenzo and O. Kornievskaia, "Efficient re-keying protocols for multicast encryption", *SCN 2002, Lecture Notes in Computer Science*, vol. 2576, pp. 119-132, 2003.
7. U. Feige, "A threshold of $\ln(n)$ for approximating set cover", *Jour. ACM*, vol. 45, pp. 634-652, July 1998.
8. A. Fiat and M. Naor, "Broadcast encryption", *Advances in Cryptology - CRYPTO'93, Lecture Notes in Computer Science*, vol. 773, pp. 480-491, 1994.
9. J.A. Garay, J. Staddon and A. Wool, "Long-lived broadcast encryption", *CRYPTO 2000, Lecture Notes in Computer Science*, vol. 1880, pp. 333-352, 2000.
10. M.R. Garey and D.S. Jonson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.
11. D. Halevy and A. Shamir, "The LCD broadcast encryption scheme", *CRYPTO 2002, Lecture Notes in Computer Science*, vol. 2442, pp. 47-60, 2002.
12. J. Lotspiech, S. Nusser and F. Prestoni, "Broadcast encryption's bright future", *IEEE Computer*, (7 pages) August 2002.
13. J.H. Ki, H.J. Kim, D.H. Lee and C.S. Park, "Efficient multicast key management for stateless receivers", *ICISC 2002, Lecture Notes in Computer Science*, vol. 2587, pp. 497-509, 2003.
14. N. Matsuzaki, T. Nakano and T. Matsumoto, "A flexible tree-based key management framework", *IEICE Trans. Fundamentals*, vol. E86-A, pp. 129-135, 2003.
15. M.J. Mihaljević, "Broadcast encryption schemes based on the sectioned key tree", *ICICS2003, Lecture Notes in Computer Science*, vol. 2836, Oct. 2003.
16. S. Mittra, "Iolus: A framework for scalable secure multicasting", *Proc. ACM SIGGCOM'97*, pp. 277-288, Sept. 1997.
17. D. Naor, M. Naor and J. Lotspiech, "Revocation and tracing schemes for stateless receivers", *CRYPTO 2001, Lecture Notes in Computer Science*, vol. 2139, pp. 41-62, 2001.

18. R. Poovendran and J. S. Baras, "An information theoretic approach for design and analysis of rooted-tree-based multicast key management schemes", *IEEE Trans. Inform. Theory*, vol. 47, pp. 2824-2834, Nov. 2001.
19. R. Poovendran and C. Bernstein, "Design of secure multicast key management schemes with communication budget constraint", *IEEE Communications Letters*, vol. 6, pp. 108-110, March 2002.
20. D. Wallner, E. Harder and R. Agee, "Key management for multicast: Issues and architectures", *RFC 2627*, <http://www.ietf.org/rfc/rfc2627.txt>
21. C.K. Wong, M. Gouda, and S.S. Lam, "Secure group communications using key graphs", *IEEE/ACM Trans. Networking*, vol. 8, pp. 16-31, Feb. 2000.

Appendix A: Sketch of Proposition 1 Proof

Recall that in SKT-A scheme there are $2^{\log_2 N - H_{0A}}$ sections in the lower layer, and each of them is controlled via the basic LSD technique [11]; the upper layer consists of only one section where CST technique [17] is employed.

Note that the re-keying of a receiver is performed via the lower layer section or the upper layer one. Accordingly, a receiver should store the keys related to LSD and CST based re-keying. A section oriented basic LSD technique requires $(H_{0A})^{1.5}$ keys, and the upper section oriented CST requires $\log_2 N - H_{0A}$ keys. So, dimension of storage@receiver overload is $O((H_{0A})^{1.5} - H_{0A} + \log_2 N)$.

Regarding the processing@receiver overload note the following. A new SEK could be delivered to the receiver employing the LSD or CST related keys. If a LSD related key is employed, the new SEK recovering at the receiver requires the processing overload proportional to H_{0A} . If a CST related key is employed, the new SEK recovering requires processing@receiver overload proportional to $\log_2 \log_2 2^{\log_2 N - H_{0A}} = \log_2(\log_2 N - H_{0A})$. So the maximum processing@receiver overload is: $O(\max\{H_{0A}, \log_2(\log_2 N - H_{0A})\}) = O(H_{0A})$.

Finally, regarding the communications overload, suppose that there are r_m revocations in the m th section, $m = 1, 2, \dots, 2^{\log_2 N - H_{0A}}$, noting that $\sum_{m=1}^{2^{\log_2 N - H_{0A}}} r_m = R$, and $\sum_{m=1}^{2^{\log_2 N - H_{0A}}} (1 - \delta_{0, r_m}) = R_{0A}$, where $\delta_{a,b}$ is a function which takes value 1 if $a = b$, and 0 otherwise. LSD based revocation within a section m requires communication overload of dimension $O(r_m)$, assuming $r_m > 0$. So, revocation of all R receivers require a communications overload of dimension $O(R)$. Also, R_{0A} revocations should be performed over the upper section employing CST, which requires additional communication overload of dimension $O(R_{0A} \log_2(2^{\log_2 N - H_{0A}}) - R_{0A} \log_2 R_{0A})$. Accordingly, dimension of the communications overload is given by $O(R + R_{0A}((\log_2 N) - H_{0A}) - R_{0A} \log_2 R_{0A})$.

Appendix B: Sketch of Proposition 2 Proof

Recall that in SKT-B scheme there are $2^{\log_2 N - H_{0B}}$ sections in the lower layer, and $2^{\log_2 N - H_{0B} - H_{1B}}$ in the middle layer: each of them is controlled via the basic LSD technique [11]; the upper layer consists of only one section where CST technique [17] is employed.

Note that the re-keying of a receiver is performed via a section within one of the tree layers, i.e., via a lower layer section or via a middle layer section or via the upper layer one. Accordingly, a receiver should store the keys related to LSD rekeying within the lower or middle layer, and CST related ones for the upper layer. Recall that the lower layer section oriented basic LSD technique requires $(H_{0B})^{1.5}$ keys, the middle layer section oriented basic LSD technique requires $(H_{1B})^{1.5}$ keys, and the upper section oriented CST requires $\log_2 N - H_{0B} - H_{1B}$ keys. So, dimension of storage@receiver overload is $O((H_{0B})^{1.5} + (H_{1B})^{1.5} - H_{0B} - H_{1B} + \log_2 N)$.

Regarding the processing@receiver overload note the following. A new SEK could be delivered to the receiver employing the LSD or CST related keys. If a LSD related key is employed, the new SEK recovering at the receiver requires the processing overload proportional to H_{0B} or H_{1B} depending whether a key from the lower or middle layer is employed. If a CST related key is employed, the new SEK recovering requires processing@receiver overload proportional to $\log_2 \log_2 2^{\log_2 N - H_{0B} - H_{1B}} = \log_2(\log_2 N - H_{0B} - H_{1B})$. So the maximum processing@receiver overload is: $O(\max\{H_{0B}, H_{1B}, \log_2(\log_2 N - H_{0B} - H_{1B})\}) = O(\max\{H_{0B}, H_{1B}\})$.

Finally, regarding the communications overload, suppose that there are r_m revocations in the m th section, $m = 1, 2, \dots, 2^{\log_2 N - H_{0B}}$, noting that $\sum_{m=1}^{2^{\log_2 N - H_{0B}}} r_m = R$, and $\sum_{m=1}^{2^{\log_2 N - H_{0B}}} (1 - \delta_{0, r_m}) = R_{0B}$, where $\delta_{a,b}$ is a function which takes value 1 if $a = b$, and 0 otherwise. LSD based revocation within a section m requires communication overload of dimension $O(r_m)$, assuming $r_m > 0$. So, revocation of all R receivers require a communications overload of dimension $O(R)$. Also, R_{0B} revocations of the sections from the lower layer should be performed within the middle layer employing middle sections oriented basic LSD approach. Employing an equivalent consideration to the above one related to the lower layer, we obtain that revocation of all R_{0B} sections in the middle layer require a communications overload of dimension $O(R_{0B})$. Additionally, R_{1B} revocations should be performed over the upper section employing CST, which requires additional communication overload of dimension $O(R_{1B} \log_2(2^{\log_2 N - H_{0B} - H_{1B}}) - R_{1B} \log_2 R_{1B})$. Accordingly, dimension of the communications overload is given by $O(R + R_{0B} + R_{1B}((\log_2 N) - H_{1B} - H_{0B}) - R_{1B} \log_2 R_{1B})$.

Leakage-Resilient Authenticated Key Establishment Protocols

SeongHan Shin, Kazukuni Kobara, and Hideki Imai

Institute of Industrial Science, The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan
shinsh@imailab.iis.u-tokyo.ac.jp, {kobara,imai}@iis.u-tokyo.ac.jp
<http://imailab-www.iis.u-tokyo.ac.jp/imailab.html>

Abstract. Authenticated Key Establishment (AKE) protocols enable two entities, say a client (or a user) and a server, to share common session keys in an authentic way. In this paper, we review AKE protocols from a little bit different point of view, i.e. the relationship between information a client needs to possess (for authentication) and immunity to the respective leakage of stored secrets from a client side and a server side. Since the information leakage would be more conceivable than breaking down the underlying cryptosystems, it is desirable to enhance the immunity to the leakage. First and foremost, we categorize AKE protocols according to how much resilience against the leakage can be provided. Then, we propose new AKE protocols that have immunity to the leakage of stored secrets from a client and a server (or servers), respectively. And we extend our protocols to be possible for updating secret values registered in server(s) or password remembered by a client.

1 Introduction

1.1 Background

Authenticated Key Establishment (abbreviated by ‘AKE’) protocols, which include Authenticated Key Agreement (AKA) and Authenticated Key Transport (AKT), are designed for two entities, say a client and a server (in case of two-party protocols), to share common session keys in an authentic way over open networks where the session keys can be used for subsequent cryptographic algorithms (e.g., symmetric key cryptosystems and message authentication codes). Since AKE protocols are crucial cryptographic primitives, they have been widely used in various protocols, such as SSH (Secure SHell) [22], SSL/TLS (Secure Socket Layer/Transport Layer Security) [14,23], and in many applications such as internet banking, electronic commerce, secure content download, secure remote access and so on. In the literature, there exist many efficient and secure AKE protocols (typical examples can be found in [18,19]) in either the random oracle model or the standard model which consider an adversary that not only can eavesdrop the communication of the entities but also can actively modify,

delete and insert messages sent between the entities of its own choice. For authentication, AKE protocols must require the involving entities to possess some information like stored secrets, passwords or public keys (or their fingerprints).

While the security of cryptosystems and protocols including AKE has been usually discussed with the assumption that stored secrets will never be revealed, we assume here the stored secrets may be leaked out. This can happen maybe due to a bug or a mis-configuration of the system. Formally,

Assumption 1 (Leakage) *Stored secrets may leak out due to accidents such as bugs or mis-configurations of the system. The source of the leakage, i.e. the bugs or the mis-configurations, will be fixed as soon as possible. But some clients continue to use the same personal information, such as passwords.*

Of course, once the bug or the mis-configuration is found, it will be patched or fixed as soon as possible and then the system will be rebuilt (if necessary). This is a common practice in Internet [10,31]. Even though the patch or the system-rebuild may remove the risk of further leakage coming from the same bug or the mis-configuration, the leaked secrets may still be abusable to intrude the newly rebuilt system or the other systems, e.g. when a client registers the same password to different servers (see Section 1.4). Thus, we think it is very important to take into account the impact of the leakage (and the burden on the client). The idea of considering leakage of stored secrets is not a new one. Already, proactive schemes [17,35], forward-secure schemes [1,2,5,11], key-insulated systems [13,24] and password-authenticated key exchange (PAKE) [3,4,6,8,15,16,19,26,27,28,29,30,32,33,36] assumed the similar situations.

Problem Setting. Let us think of secrets stored in devices in the model of proactive schemes, forward-secure schemes and key-insulated systems where the secrets should be updated or refreshed regularly in a predetermined time period or at a time when a client (or a server) notices the leakage of stored secrets. Specifically, proactive schemes [17,35] improve threshold schemes by allowing multiple leakage of secrets, limiting only the number of simultaneous leakages. While forward-secure schemes [1,2,5,11] evolve secrets at the end of each time period, key-insulated systems [13,24] update secrets with update information coming from TRM (Tamper-Resistant Modules). All of them can minimize the impact of the leakage, but not completely prevent the damage. In addition, it takes some time from when stored secrets leaked out until the client (or the server) can realize the fact and then the secrets are updated by new ones. Within the term for realizing the fact or the time period for updating, an adversary who obtained the secrets can break its security in a limited time period. Bringing this problem into AKE protocols, which use *only stored secrets*, may end up with the same result as above. That's the reason why authentication totally depends on stored secrets so that leakage of the secrets is directly connected to impersonating the victimized entity. For the countermeasure, there exist AKE protocols using a password, without TRM. However, most of PAKE protocols requiring *only a password* on client's side can provide a solution against leakage of stored secrets

from a client, not a server. Thus, it is desirable to provide immunity to the leakage of stored secrets from a client and a server, respectively. More detailed discussion will be provided through this section. From now on, we focus on AKE protocols using password where the password naturally takes a major role for authentication.

1.2 Classification of AKE Protocols Using Password

In the literature, various AKE protocols have been proposed so far which could be divided by what is used to authenticate entities. Here, we classify them according to the types of information a client needs to possess.

At first, let us start by categorizing the *types of information* to be possessed by a client as follows. (i) Human-Memorable Secret (HMS): A secret, which is remembered and typed in by a client, such as a password. (ii) Stored Secret (SS): Secrets stored in a client's machine, in a memory card or somewhere that is not protected with perfect tamper-resistant modules. It may be merely secret values, a signing key of a digital signature scheme, a decryption key of a public key cryptosystem and/or a common key of a symmetric key cryptosystem. (iii) Public Information (PI): Public information, such as a verification key of a digital signature scheme, an encryption key of a public key cryptosystem or their fingerprints. While anyone can get the public information, its validity must be verified at their own responsibility.

Additionally, we assume the followings on the HMS and the SS.

Assumption 2 (Short but Secure to On-line Attacks) *The size of the human-memorable secret is short enough to memorize, but large enough to avoid on-line exhaustive search. This means the secret may be vulnerable to off-line exhaustive search.*

The on-line attack is a series of exhaustive search for a secret performed on-line where adversaries are willing to sieve out secret candidates one by one running an authentication protocol with the target entity (usually, server). In contrast, the off-line attack is performed off-line massively in parallel with recorded transcripts of a protocol. While on-line attacks are applicable to all of the protocols using password equally, they can be prevented by letting a server take appropriate intervals between invalid trials. But, we cannot avoid off-line attacks by such policies, mainly because the attacks are performed off-line and independently of the server. As a result, off-line attacks are critical to most of the protocols using human-memorable passwords.

Assumption 3 (No Perfect TRM) *TRM (Tamper-Resistant Modules) used to store the secrets are not perfectly free from bugs and mis-configurations.*

With the above types of information, we differentiate previous AKE protocols¹. At first, we list up typical AKE protocols using HMS and explain how they work briefly. (We ignore protocols, which are vulnerable to off-line attacks as they are, such as CHAP [20], IPsec with pre-shared secret [21] and so on.)

¹ A more detailed description of previous AKE protocols will be given in [38].

SSL/TLS and SSH. We show two AKE protocols of SSL/TLS and SSH. (For formal description of the following protocols, refer to SSL/TLS [14,23] and SSH [22].)

1. Password-based User Authentication over a Secure Channel: In this scheme, a client establishes a secure connection to a server, and then sends the client's password for authentication through the secure connection. The server verifies the given password in the same way as the usual password verification procedure. Note that the server needs to store (a hashed value of) the password.
2. Public-Key based User Authentication with a Password-Protected Secret-Key: A server verifies a client's secret key using a challenge-response protocol. In addition to that, the client stores the secret key in encrypted form with his password.

Password-Authenticated Key Exchange (PAKE) Protocols. PAKE protocols are designed for entities to share a fresh session key (to be secure against off-line attacks) by using *only a pre-shared human-memorable password*, which may be exhaustible with off-line attacks but not with on-line attacks.

A brief sketch of PAKE protocols, which only rely on a password, is given as follows. Both a client and a server share the same password in advance. For authentication and key exchange, they run a PAKE protocol using the shared password (or a hashed value of it). If their inputs coincide with each other, they can obtain the same value that is used to generate a session key for secure channels. Otherwise, they get distinct values which are hard to guess each other. Thus, no adversary can intrude in the middle of them or impersonate one entity to the other.

Up to now, a variety of studies on PAKE protocols [3], [4], [6], [8], [15], [16], [19], [26], [27], [28], [29], [30], [32] have appeared in the literature. In PAKE protocols, a client keeps in mind his password whereas the counterpart server should have its verification data that is used to verify the client's knowledge of the password. That means leakage of stored secrets (that is, verification data) from the server makes possible off-line dictionary attacks for an adversary.

Threshold-PAKE (T-PAKE) Protocols. In order to prevent the leakage of stored secrets from a server, [33,36] proposed T-PAKE protocols where a client's password or verification data is not stored in a single server but rather shared among a set of servers using a secret sharing scheme. Since only a certain threshold of servers can reconstruct the client's password or verification data in the authentication phase, the leakage of stored secrets from any number of servers smaller than the threshold doesn't help an adversary to perform off-line attacks.

1.3 Evaluation by Immunity to the Leakage

As mentioned in Section 1.1, we consider the situation that stored secrets from a client and a server may leak out due to a bug or a mis-configuration of the

Table 1. Attack and security levels.

Attack levels	On-line attacks	Off-line attacks
Security levels		
Strongly secure ^{*1} (○)	Secure	Secure
Weakly secure ^{*2} (△)	Secure	Insecure

*1: An AKE protocol using password is said to be strongly secure (denoted by ○), if the protocol can be tolerant against both on-line and off-line attacks.

*2: An AKE protocol using password is said to be weakly secure (denoted by △), if the protocol can be tolerant against on-line but not off-line attacks.

system. Before evaluating previous AKE protocols using password according to immunity to the leakage of stored secrets, we divide security levels into two cases with respect to whether an AKE protocol can maintain its security (with the client's password unknown to an adversary) against on-line and off-line attacks and then summarize them in Table 1.

With these security levels, we summarize comparative results in Table 2 about whether a client (or a server) can remain resistant against on-line and off-line attacks *even after* stored secrets from the client (or the server) are leaked out to an adversary, respectively. For simplicity, we evaluate immunity to the leakage of each class of AKE protocols presented in Section 1.2.

As shown in the table, PAKE protocols just require that a client keep in mind his password while the counterpart server should have its verification data associated with the password. Consequently, if stored secrets in the server are leaked out, an adversary who gets them can retrieve the original password through off-line attacks, simply by verifying password candidates one by one using the verification data. As a countermeasure to the leakage from server, [33,36] provided a solution in which n ($n > 1$) servers share verification data (or verification function) using a secret sharing scheme and the threshold of servers participate in the protocol to authenticate a client. That is, the leakage of stored secrets from any number of servers smaller than the threshold does not make off-line attacks possible. However, the client's password can be retrieved if stored secrets from the threshold or more than the threshold of servers are leaked out. In a word, it is impossible for PAKE (T-PAKE) protocols using only HMS (and PI) to achieve strong security against the leakage from server(s).

Fact 1 (Impossibility of Strong Security in PAKE and T-PAKE) *PAKE (T-PAKE) protocols, requiring only HMS (and PI) as clients' possessions, cannot achieve the strong security against the leakage from server(s). For any such a protocol, an adversary can perfectly simulate the protocol using the leaked secrets from server(s) so that he/she can try the password candidates off-line in parallel.*

SSL/TLS and SSH in the password-based user authentication mode make a server keep a hashed value of a client's password. As a matter of course, leakage of stored secrets from the server results in revealing the password through off-line attacks. In the SSL/TLS and SSH of the public-key based user authentication

Table 2. Comparison of AKE protocols using password.

Protocols	Client's possessions			Immunity to leakage	
	HMS	SS	PI	from Client	from Server
PAKE* ¹	✓			○	△
Our Proposals	✓	✓		○	○
SSL/TLS* ² , SSH* ² , T-PAKE	✓		✓	○	△* ⁴
SSL/TLS* ³ , SSH* ³	✓	✓	✓	△	○

*1: Most PAKE protocols, being secure against server compromise^a, which hold clients' verification data

*2: Key-establishment part of SSL/TLS and SSH in the password-based user authentication mode^b

*3: Key-establishment part of SSL/TLS and SSH in the public-key based user authentication mode with a password-protected secret-key^c

*4: T-PAKE protocols [33,36] have the immunity up to its threshold of servers.

^a Throughout this paper, we use the terminology of 'leakage' rather than 'compromise'.

^b More specifically, password authentication after the server authentication in SSL/TLS or the password authentication in SSH.

^c More specifically, mutual authentication in SSL/TLS, RSA authentication in SSH protocol version 1 or public key authentication in SSH protocol version 2.

mode with a password-protected secret-key, leakage from a client can prevent an adversary, who is willing to get the client's password, from obtaining the password through only on-line attacks, but not off-line attacks.

As a consequence, Table 2 indicates that the existing AKE protocols using password are vulnerable to the leakage from either client or server. That means any of the AKE protocols (except our protocols) doesn't provide immunity to the leakage of stored secrets from *client and server*, respectively. Remind that AKE protocols, which use only stored secrets, can minimize the impact of the leakage by updating or refreshing the secrets, but not completely prevent the damage.

1.4 A Realistic, but Critical, Problem of AKE Protocols Using Password

Are all the existing AKE protocols using password really secure *in the real world*? Instead of answering to the question, we take for an example a very compelling but critical situation in the real world.

Let us think of an ordinary client who would connect with several disparate servers, each requiring a password, over networks for internet banking, internet shopping, internet auction, ftp servers, electronic voting and so on. As of now, all of the AKE protocols *implicitly* have the assumption that the client registers *information-theoretically independent passwords* corresponding to different servers. Remember that password can be defined as human beings have some-

thing *memorable* usually in size of 6-8 characters (including numbers). Ironically, how many passwords can we remember? 10 or 20? Of course, it depends on the individual. Here, we have another assumption as follows:

Assumption 4 (One Memorized Secret) *A client remembers only one human-memorable secret, i.e. one password, even if he/she communicates with several different servers. That means the client use the same password to a distinct kind of servers, not sharing any secret information one another.*

Under the Assumption 4 in the multiple server scenario, we have to take into consideration the impact on other servers after the leakage of stored secrets from one server in the real world.

Definition 1 (Impact after the Leakage from One Server) *An AKE protocol using password, where there is no impact on other servers after the leakage of stored secrets from one server, is said to be desirable in the sense that an adversary, after obtaining verification data associated with a client's password from one server, cannot retrieve the password that makes possible to impersonate the client to other servers of the Assumption 4. That is, the password is completely protected against off-line attacks even if the adversary can get some verification data from servers.*

Of course, a client may change his password *instantly* to all servers at a time when he comes to know that stored secrets from one of the servers are revealed out. However, it triggers the burden on the client.

Motivation. The motivation of this paper is on how to design an AKE protocol that has immunity to the leakage of stored secrets from a client and servers, respectively, under the Assumption 4. That means the client need not change his password, *even if* stored secrets are leaked out from either the client or servers. However, we can easily deduce the following fact that there exists no AKE protocol, which is immune to the leakage from a client and servers *simultaneously*.

Fact 2 (Impossibility of Perfect Security) *Any AKE protocol cannot achieve the strong security against the leakage from both a client and servers simultaneously. If an adversary obtains stored secrets from both a client and servers at the same time, he/she can perfectly simulate the protocol using the leaked secrets. Thus the adversary can try the password candidates off-line in parallel.*

This fact motivates us to achieve the next highest goal, i.e. the strong security against the leakage from a client and servers, respectively. Notice that our protocol is *not* a kind of PAKE protocols, but a new one that requires *one password* and *secret values* on client's side.

1.5 Our Contributions

In this paper, we propose new AKE protocols that are immune to the leakage of stored secrets from *both a client and servers* respectively, as long as the leakages are not simultaneous, where the client keeps *one password* in his mind and stores *secret values* in devices. Specifically speaking, a client registers a partial secret value (which is not a share itself) of one password to a different kind of servers by means of a secret sharing scheme. The protocol of Section 2.2 is a generalized version in which the number of servers is fixed in advance whereas the second in Section 2.3 can be readily applied to the real world, simply because the latter considers synchronization between a client and one of the servers for registering a secret value. That means the client can compute a secret value with the same password at any time when needed to register to a necessary server without restricting the number of servers. In our protocols, an adversary obtaining stored secrets after the leakage from all of the servers (the client has been communicating with) cannot find out the password. Also, an adversary getting stored secrets after the leakage from the client cannot sieve out the password. More interestingly, the password remains *information-theoretically secure* even if the leakage of stored secrets from the client and servers happens, respectively.

In addition to that, our protocols have the following advantages: (1) the proposed protocols can be constructed with small modifications of the widely used Diffie-Hellman key exchange protocol [12]. (2) the proposed protocols have a formal validation of security in the standard model (instead of the random oracle model [9]) under the assumption that DDH (Decisional Diffie-Hellman) problem is hard and MACs are selectively unforgeable against partially chosen message attacks (which is a weaker notion than being existentially unforgeable against chosen message attacks).

Then, we extend our protocol of Section 2.2 to two protocols where one enables a client to update each of the secret values registered in different servers without changing his password (which might be remembered with considerable effort)² and the other enables a client to change his password with a new password while updating each of the secret values in different servers.

For better understanding, it may be helpful to state about what is different between our approach and T-PAKE protocols [33,36]. The main difference is that [33,36] cannot preserve its security (a client's password) against off-line attacks if stored secrets from the threshold or more than the threshold of servers would be revealed, whereas ours can maintain it *even after* stored secrets from all of the servers (a client is communicating with) would be revealed out. That's the reason why [33,36] proposed T-PAKE protocols in order to protect a client's password from the leakage of stored secrets in a server where verification data (or function) associated with the password is distributed by a set of servers.

² This additional function is useful when we consider a situation where one administrator of servers resigns with secret values (associated with clients' passwords) in the server. However, recall that the frequent change of passwords rather increases the risk of password to be lost and cracked, simply because people tend to write it down on somewhere.

Contrarily, our protocols distribute secret values computed with the password by the client himself. Accordingly, if stored secrets from the threshold or more than the threshold of servers are leaked out in [33,36], clients' passwords can be retrieved by an adversary which affects on different servers that don't share any secret information one another under the Assumption 4. Besides, both the communication and the computation complexity of [33,36] are by far larger than ours. And, applications of [33,36] are restricted since a certain threshold of servers must take part in the protocol for authentication.

1.6 Organization

This paper is organized as follows: In Section 2, we propose new AKE protocols that have immunity to the leakage of stored secrets from not only a client but also servers, respectively. Section 3 shows how our protocols can remain resistant against off-line attacks even after the leakage of stored secrets from the client and servers, respectively. Then, we extend the proposed protocols in Section 4.

2 Our Proposals: Leakage-Resilient AKE Protocols

2.1 Scenario

Here, we consider the following scenario that there are $n - 1$ ³ disparate kinds of servers communicating with a client, who wants to use *one password* and *secret values* to produce cryptographically secure (or, high entropy) session keys with different servers at any time.

Our protocols are defined over a finite cyclic group $\mathcal{G} = \langle g \rangle$ where $|\mathcal{G}| = q$ and q is a large prime (or, a positive integer divisible by a large prime). While \mathcal{G} can be a group over an elliptic curve, we assume that \mathcal{G} is a prime order subgroup over a finite field \mathbb{F}_p . That is, $\mathcal{G} = \{g^i \bmod p : 0 \leq i < q\}$ where p is a large prime number, q is a large prime divisor of $p - 1$ and g is an integer such that $1 < g < p - 1$, $g^q \equiv 1$ and $g^i \not\equiv 1$ for $0 < i < q$. A generator of \mathcal{G} is any element in \mathcal{G} except 1. In the aftermath, all the subsequent arithmetic operations are performed in modulo p , unless otherwise stated. Both g and h are two generators of \mathcal{G} so that its DLP (Discrete Logarithm Problem), i.e. calculating

$$a = \log_g h, \tag{1}$$

should be hard for each entity. Both g and h may be given as system parameters or chosen with an initialization phase between entities.

The protocols consist of the following four phases: an initialization phase, a secrecy amplification phase, a verification phase and a session-key generation

³ In case of two-party protocols, n becomes 2. As our protocols also satisfy the two-party case, we set up n ($2 \leq n < q$), at the same time, in order to consider the multiple server scenario of Assumption 4.

phase. In the initialization phase, a client registers each of the secret values computed by himself to different servers. Then, he stores the corresponding secret values in devices such as smart cards or computers and keeps only one password in mind. In the secrecy amplification phase, secrecy of a weak secret, i.e. a human-memorable password that may be vulnerable against off-line attacks, is amplified to a strong secret (we call it a keying material) that is secure even against off-line attacks. In the verification phase, both client and server can confirm whether or not they share the same keying material using a challenge-response protocol with the keying material as its key. In the session-key generation phase, a session key is generated using the keying material.

2.2 A Leakage-Resilient AKE Protocol

We describe a construction for a leakage-resilient AKE protocol which is illustrated in Fig. 1. The key idea behind our protocol is that a client can generate n shares of his password, where each of the $n - 1$ shares is used for registering a secret value to the corresponding server and the remaining one share (not itself) is stored in his devices in the initialization phase, only by inputting the password (as a secret value) into (n, n) -threshold secret sharing scheme of [7,37].

[Initialization] A client C , included in n entities, is willing to register each of the secret values generated by one password pw to the respective $n - 1$ different server S_i ($1 \leq i \leq n - 1$). For simplicity, we assign the servers consecutive integer $1 \leq i \leq n - 1$ where i can be regarded as each server's ID and n as the client's ID. First and foremost, the client picks a random polynomial $p(x)$ of degree $n - 1$ with coefficients also randomly chosen in $(\mathbb{Z}/q\mathbb{Z})^*$:

$$p(x) = \sum_{j=0}^{n-1} \alpha_j \cdot x^j \bmod q \quad (2)$$

and sets $\alpha_0 = pw$ ⁴ where pw is the client's password. After computing the respective shares $p(i)$ ($1 \leq i \leq n - 1$) with the above polynomial, he registers securely each of the secret values $h^{p(i) \cdot \lambda_i}$ to the corresponding server S_i ($1 \leq i \leq n - 1$) as follows:

$$S_i \leftarrow h^{p(i) \cdot \lambda_i}, \text{ where } \lambda_i = \prod_{k=1, k \neq i}^n \frac{k}{k - i} \bmod q \quad (3)$$

where $p(i)$ is a share of (n, n) -threshold secret sharing scheme and λ_i is a Lagrange coefficient. Note that share $p(n)$, which is for the client, is never registered to any server. Then, the client *just stores the corresponding secret values*

⁴ Instead of pw , a hashed value of the password can be used. In either case where both have the same entropy, it doesn't affect on the security.

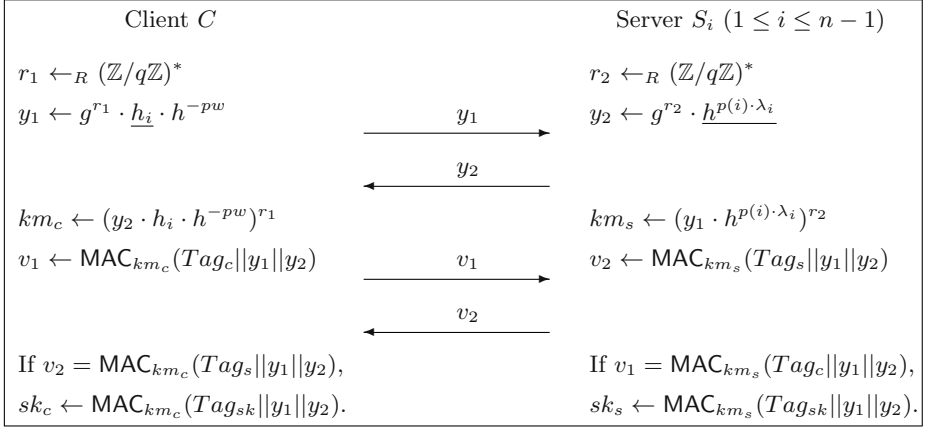


Fig. 1. A leakage-resilient AKE protocol. The underlined values represent stored secrets of client and server, respectively.

h_i ($1 \leq i \leq n-1$) to the servers S_i in devices, such as smart cards or computers, which may happen to leak the secrets h_i and *keeps his password pw in mind*.

$$h_i \leftarrow h^{\sum_{l=1, l \neq i}^n p(l) \cdot \lambda_l}. \quad (4)$$

Of course, all the other (intermediate) values should be deleted from the devices.

[Secrecy Amplification] When the client C wants to share a session key with one of the servers S_i ($1 \leq i \leq n-1$), he chooses a random number $r_1 \leftarrow_R (\mathbb{Z}/q\mathbb{Z})^*$. Then, the client sends y_1 to server S_i , after calculating $y_1 \leftarrow g^{r_1} \cdot h_i \cdot h^{-pw}$ using the corresponding secret value h_i to the server and his password pw that is partially shared with the server. The server S_i also calculates $y_2 \leftarrow g^{r_2} \cdot h^{p(i) \cdot \lambda_i}$ with a random number $r_2 \leftarrow_R (\mathbb{Z}/q\mathbb{Z})^*$ and its secret value $h^{p(i) \cdot \lambda_i}$ (partial secret information about the password) registered by the client in the initialization phase, and then transmits it to the client. On both sides, the client's keying material becomes $km_c \leftarrow (y_2 \cdot h_i \cdot h^{-pw})^{r_1}$ and the server's one becomes $km_s \leftarrow (y_1 \cdot h^{p(i) \cdot \lambda_i})^{r_2}$.

Only if the client uses the right password pw and the corresponding secret value h_i to server S_i and the server S_i uses the right secret value $h^{p(i) \cdot \lambda_i}$, both of them can share the same keying material that is obtained by Lagrange interpolation:

$$km_c = (y_2 \cdot h_i \cdot h^{-pw})^{r_1} = \left(g^{r_2} \cdot h^{p(i) \cdot \lambda_i} \cdot h^{\sum_{l=1, l \neq i}^n p(l) \cdot \lambda_l} \cdot h^{-pw} \right)^{r_1} = g^{r_2 \cdot r_1}, \quad (5)$$

$$km_s = \left(y_1 \cdot h^{p(i) \cdot \lambda_i} \right)^{r_2} = \left(g^{r_1} \cdot h^{\sum_{l=1, l \neq i}^n p(l) \cdot \lambda_l} \cdot h^{-pw} \cdot h^{p(i) \cdot \lambda_i} \right)^{r_2} = g^{r_1 \cdot r_2}. \quad (6)$$

Otherwise guessing the other's keying material is hard due to the DLP (see [38]). Also, adversaries cannot determine the correct password of the client through

off-line attacks since they don't know the client's random number r_1 chosen at the time and the secret value h_i corresponding to sever S_i , both of which are required to narrow down the password pw .

This phase ends up with only one pass in parallel since both y_1 and y_2 can be calculated and sent independently (where $g^{r_1} \cdot h_i$ and y_2 are pre-computable). Additionally, the implementation cost of this phase is very low because it can be simply obtained from a small modification of widely used Diffie-Hellman key exchange protocol [12]. That's why $h^{-p(i) \cdot \lambda_i} = h_i \cdot h^{-pw}$.

[Verification] In this phase, a pair of entities can verify whether they share the same keying material or not with a challenge-response protocol using the keying material calculated in the secrecy amplification phase.

The client and the server calculate $v_1 \leftarrow \text{MAC}_{km_c}(\text{Tag}_c || y_1 || y_2)$ and $v_2 \leftarrow \text{MAC}_{km_s}(\text{Tag}_s || y_1 || y_2)$, respectively, using a MAC generation function $\text{MAC}_k(\cdot)$ with the keying materials as its key k . Both Tag_c and Tag_s are pre-determined distinct values, e.g. $\text{Tag}_c = (ID_c || ID_s || 00)$ and $\text{Tag}_s = (ID_c || ID_s || 01)$ where ID_c and ID_s are IDs of the client and the server respectively. Then, they exchange v_1 and v_2 each other, before verifying $v_2 = \text{MAC}_{km_c}(\text{Tag}_s || y_1 || y_2)$ and $v_1 = \text{MAC}_{km_s}(\text{Tag}_c || y_1 || y_2)$ on both sides. If at least one of them does not hold, the corresponding entities wipe off all the temporal data including the keying materials, and then close the session. Otherwise they proceed to the session-key generation phase.

Adversaries can try off-line attacks for the keying material using $\{(\text{Tag}_c || y_1 || y_2) \text{ and } v_1\}$ or $\{(\text{Tag}_s || y_1 || y_2) \text{ and } v_2\}$. The success probability achieved within a polynomial time t can be negligible if a strong secret can be shared in the secrecy amplification phase and an appropriate MAC generation function, whose keys are unguessable, is used.

[Session-Key Generation] If the above verification phase succeeds in, the entities generate their session keys using the verified keying materials as follows:

$$sk_c \leftarrow \text{MAC}_{km_c}(\text{Tag}_{sk} || y_1 || y_2) \quad (7)$$

$$sk_s \leftarrow \text{MAC}_{km_s}(\text{Tag}_{sk} || y_1 || y_2) \quad (8)$$

where Tag_{sk} is a pre-determined distinct value from both Tag_c and Tag_s , e.g. $\text{Tag}_{sk} = (ID_c || ID_s || 11)$. The generated session keys are used for their subsequent cryptographic algorithms.

The requirement for the MAC generation function in this phase and the previous phase is $\epsilon_{mac}(k_2, t, i)$ can be negligibly small for a practical security parameter k_2 and i (this is a polynomial of k_2). That's the reason why if adversaries cannot forge a MAC corresponding to $(\text{Tag}_{sk} || y_1 || y_2)$ and km_c or km_s with significant probability, they cannot obtain any information of the session key. This requirement can be satisfied by using a universal one-way hash function [34] or by using a practical MAC generation function, such as HMAC-SHA-1 [25] (and even KeyedMD5), since any effective algorithms have not been known so far to

make $\epsilon_{mac'}(k_2, t, i)$ non-negligible where $\epsilon_{mac'}(k_2, t, i)$ is larger than or equal to $\epsilon_{mac}(k_2, t, i)$.

2.3 A More Practical Leakage-Resilient AKE Protocol

The proposed protocol in Section 2.2 deployed a (n, n) -threshold secret sharing scheme, in order to generate $n - 1$ secret values with each registered to $n - 1$ servers respectively. That is, a client should determine the number of different servers in advance and register each of the secret values to the servers all at once. When it comes to the real world, it is desirable that a client be able to choose among a different kind of servers at his own will. Although a client in the protocol of Section 2.2 can choose $n - 1$ different servers, we show how to apply the proposed protocol to the case where the client can compute a secret value (from one password) at any time when needed. This approach will lead the protocol of Section 2.2 to be more simpler in the initialization phase, just by replacing (n, n) -threshold secret sharing scheme with $(2, 2)$ -threshold one.

[Initialization] A client C is willing to register a secret value generated by one password pw to one of different servers S_i where i can be regarded as each server's ID. Every time when needed to register a secret value to a server, the client picks a distinct random polynomial $p_i(x)$ (for the respective server S_i) of degree 1 with coefficient α_{i1} randomly chosen in $(\mathbb{Z}/q\mathbb{Z})^*$:

$$p_i(x) = \sum_{j=0}^1 \alpha_{ij} \cdot x^j = \alpha_{i0} + \alpha_{i1} \cdot x \bmod q \quad (9)$$

and sets $\alpha_{i0} = pw$ where pw is the client's password. After computing a share $p_i(1)$ with the above polynomial, he registers securely a secret value $h^{p_i(1) \cdot \lambda_1}$ to one of different servers S_i as follows:

$$S_i \leftarrow h^{p_i(1) \cdot \lambda_1}, \text{ where } \lambda_1 = 2 \bmod q \quad (10)$$

where $p_i(1)$ is a share of $(2, 2)$ -threshold secret sharing scheme for the server S_i and λ_1 is a Lagrange coefficient. Note that share $p_i(2)$ is for the client. Then, the client just stores *the corresponding secret value* h_i in devices and keeps *his password* pw in mind.

$$h_i \leftarrow h^{p_i(2) \cdot \lambda_2}, \text{ where } \lambda_2 = -1 \bmod q. \quad (11)$$

The rest phases of this protocol are as same as those of Section 2.2.

3 Security

This section shows the security of password in Section 2.2 against off-line attacks after the leakage of stored secrets from a client and servers, respectively. And the

security of password in Section 2.3, which is a case of $n = 2$ in (n, n) -threshold secret sharing scheme, inherits from Section 3 straightforwardly. Moreover, the security against on-line and off-line attacks of the below adversary can be proven in the standard model as Theorem 2. (For formal security proof, refer to [38].)

In the security model of our protocol, we consider a far more powerful adversary who has ability to not only eavesdrop, modify and delete the messages exchanged by entities, but also to insert messages of its own choice. This adversarial power is modeled by giving the adversary oracle access to the instances of our protocol. In addition, the adversary is given access to a Leak oracle that simulates Assumption 1. That is, Leak oracle accepts an entity ID and then reveals the corresponding stored secrets. However, this oracle does not reveal stored secrets of its partner at the same time, because of Fact 2.

3.1 Security of Password against the Leakage

The primary goal of an adversary after obtaining stored secrets from a client and servers, respectively, is to perform off-line exhaustive search for the client's password that makes possible to impersonate the client to other servers under the Assumption 4.

Theorem 1 *The password in our protocol of Section 2.2 remains information-theoretically secure against off-line attacks after the leakage of stored secrets from the client C and $n - 1$ servers S_i ($1 \leq i \leq n - 1$), respectively. Even if an adversary obtains stored secrets from the Leak oracle, she cannot retrieve the client's original password through off-line exhaustive search that is the best attack for the adversary.*

Proof. When an adversary gets secrets stored in devices from the client C and $n - 1$ servers S_i ($1 \leq i \leq n - 1$) respectively, what she wants to know is the client's password pw or a value associated with the password

$$h^{pw} = h^{\sum_{m=1}^n p(m) \cdot \lambda_m}. \quad (12)$$

Only if the above value h^{pw} is computed, the adversary can narrow down the original password by checking possible password candidates with equation (12) one by one (through off-line exhaustive search). In order to simplify the proof, let us fix $n = 5$.

First, we think of the security of password against an adversary who obtains stored secrets h_i ($1 \leq i \leq 4$) of the client C and is trying to deduce $h^{\sum_{m=1}^5 p(m) \cdot \lambda_m}$ for the client's password pw . Below is the exponent part of h_i

$$\begin{bmatrix} \log_h h_1 \\ \log_h h_2 \\ \log_h h_3 \\ \log_h h_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p(1) \cdot \lambda_1 \\ p(2) \cdot \lambda_2 \\ p(3) \cdot \lambda_3 \\ p(4) \cdot \lambda_4 \\ p(5) \cdot \lambda_5 \end{bmatrix}. \quad (13)$$

Equation (13) means that the secrets h_i ($1 \leq i \leq 4$) don't reveal any information about the password pw , simply because each row contains 4 shares (the number of shares needed for the client's password is more than that of h_i by one share) and each exponent part of h_i is linearly independent one another. That is the adversary cannot compute $h^{\Sigma_{m=1}^5 p(m) \cdot \lambda_m}$ with stored secrets h_i .

Second, we think of the security of password against an adversary who obtains stored secrets $h^{p(i) \cdot \lambda_i}$ of all the servers S_i ($1 \leq i \leq 4$) and is trying to deduce $h^{\Sigma_{m=1}^5 p(m) \cdot \lambda_m}$ for the client's password pw . Below is the exponent part of $h^{p(i) \cdot \lambda_i}$:

$$\begin{aligned} \log_h S_1 &\rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} p(1) \cdot \lambda_1 \\ p(2) \cdot \lambda_2 \\ p(3) \cdot \lambda_3 \\ p(4) \cdot \lambda_4 \\ p(5) \cdot \lambda_5 \end{bmatrix} . \end{aligned} \quad (14)$$

Intuitively, the number of shares included in $h^{\Sigma_{m=1}^5 p(m) \cdot \lambda_m}$ is one more than that of $h^{p(i) \cdot \lambda_i}$ ($1 \leq i \leq 4$), since each row only contains one share of (5, 5)-threshold secret sharing scheme. Although the adversary gathers all of the secret values from servers S_i ($1 \leq i \leq 4$), the number of shares is 4. That means the password is information-theoretically secure as a secret value of (5, 5)-threshold secret sharing scheme. \square

Theorem 2 (Indistinguishability of sk) *Suppose the following adversary \mathcal{A} , which accepts a challenge transcript (that may be obtained by eavesdropping a protocol, impersonating a partner or intruding in the middle of the target entities), and then asks q_{ex} , q_{se} , q_{re} and q_{le} queries to the Execute, Send, Reveal, Leak oracles respectively, and finally is given sk_x by Test_{sk} oracle where sk_x is either the target session key or not with the probability of $1/2$. Then $\text{Adv}_{\mathcal{A}}^{\text{ind}_{sk}}$, the advantage of adversary \mathcal{A} to distinguish whether sk_x is the target session key or not in a polynomial time t , is upper bounded by*

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^{\text{ind}_{sk}} &\leq \varepsilon_{\text{mac}}(k_2, t, q_{se} + 2q_{ex} + q_{re} + 2) + 2(q_{se} + q_{ex} + 1) \cdot \varepsilon_{\text{ddh}}(k_1, t) \\ &\quad + \frac{2(q_{se} + 1)}{N} + \frac{2(2q_{se} + q_{ex} + 1)}{|\mathcal{G}|} \end{aligned} \quad (15)$$

where both k_1 and k_2 are the security parameters.

4 Extensions

It is reasonable that a client has control of *each of the secret values* registered in a different kind of servers and of *password* kept in his mind, regularly or irregularly. Here, we provide two extended versions of Section 2.2, simply by using a proactive threshold scheme [35] in which there is a basic assumption that an adversary who gets stored secrets from a server cannot take the update

information. One is for the secret-values update which enables a client to update each of the secret values stored in different servers without changing his password. And the other is for the password update which enables a client to change his password with a new one while updating each of the secret values in different servers. In the point of view of updating stored secrets, our approach is similar to those of key-insulated systems [13] and intrusion-resilient signatures [24]. However, the main difference is that we don't use TRM (Tamper-Resistant Modules) to produce update information, which can be computed by the client himself in our protocol. We omit two versions of Section 2.3, whose extensions can be readily shown in the same way of Section 4.

[Secret-Values Update (for Proactive Security)] When a client C , included in n entities, wants to update each of the secret values which has been registered to the respective $n - 1$ different servers S_i ($1 \leq i \leq n - 1$) with new ones (to be generated by the same password pw), he picks another random polynomial $p'(x)$ of degree $n - 1$ with coefficients randomly chosen in $(\mathbb{Z}/q\mathbb{Z})^*$:

$$p'(x) = \sum_{j=1}^{n-1} \beta_j \cdot x^j \bmod q \quad (16)$$

and sets $\beta_0 = 0$. After computing the respective shares $p'(i)$ ($1 \leq i \leq n - 1$) with the above polynomial, the client transmits securely each of the new secret values $h^{p'(i) \cdot \lambda_i}$ to the corresponding server S_i ($1 \leq i \leq n - 1$) as follows:

$$S_i \leftarrow h^{p'(i) \cdot \lambda_i}, \text{ where } \lambda_i = \prod_{k=1, k \neq i}^n \frac{k}{k - i} \bmod q \quad (17)$$

where $p'(i)$ is a new share of (n, n) -threshold secret sharing scheme and λ_i is a Lagrange coefficient. Consequently, each server S_i can produce an updated secret value $h^{(p(i)+p'(i)) \cdot \lambda_i} = h^{p(i) \cdot \lambda_i} \cdot h^{p'(i) \cdot \lambda_i}$ with multiplying the previous secret value $h^{p(i) \cdot \lambda_i}$ by a new one $h^{p'(i) \cdot \lambda_i}$. Note that share $p'(n)$, which is for the client, is never registered to any server. Then, the client also updates and stores *the corresponding secret values* $h_i' = h^{\sum_{l=1, l \neq i}^n (p(l)+p'(l)) \cdot \lambda_l}$ ($1 \leq i \leq n - 1$) in devices and keeps *the same password* pw in mind.

$$h_i' \leftarrow h_i \cdot h^{\sum_{l=1, l \neq i}^n p'(l) \cdot \lambda_l}. \quad (18)$$

Of course, the client doesn't need to update secret values stored in different servers S_i ($1 \leq i \leq n - 1$) *simultaneously*. That means he can update each of the secret values in servers at any time, only if the client chooses a different random polynomial every time.

[Password Update] If a client C wants to change his password pw with a new one pw_{new} while updating each of the secret values registered to the respective $n - 1$ different servers S_i ($1 \leq i \leq n - 1$), he follows the above secret-values update in the same way except that the client picks another random polynomial $p''(x)$ of degree $n - 1$ with coefficients randomly chosen in $(\mathbb{Z}/q\mathbb{Z})^*$:

$$p''(x) = \sum_{j=0}^{n-1} \gamma_j \cdot x^j \bmod q \quad (19)$$

and sets $\gamma_0 = -pw + pw_{new}$ where pw_{new} is the client's new password.

Acknowledgements

The authors would like to thank anonymous referees for useful comments.

References

1. M. Abdalla, S. Miner, and C. Namprepmpre. Forward-Secure Threshold Signature Schemes. In *Proc. of Topics in Cryptology (CT-RSA 2001)*, LNCS 2020, pages 441-456. Springer-Verlag, 2001.
2. R. Anderson. Two Remarks on Public Key Cryptology. *Technical Report*, No. 549, University of Cambridge, December 2002.
3. E. Bresson, O. Chevassut, and D. Pointcheval. Group Diffie-Hellman Key Exchange Secure against Dictionary Attacks. In *Proc. of ASIACRYPT 2002*, LNCS 2501, pages 497-514. Springer-Verlag, 2002.
4. S. M. Bellare and M. Merritt. Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In *Proc. of IEEE Symposium on Security and Privacy*, pages 72-84, 1992.
5. M. Bellare and S. Miner. A Forward-Secure Digital Signature Scheme. In *Proc. of CRYPTO '99*, LNCS 1666, pages 431-448. Springer-Verlag, 1999.
6. V. Boyko, P. MacKenzie, and S. Patel. Provably Secure Password-Authenticated Key Exchange using Diffie-Hellman. In *Proc. of EUROCRYPT 2000*, LNCS 1807, pages 156-171. Springer-Verlag, 2000.
7. G. R. Blakley. Safeguarding Cryptographic Keys. In *Proc. of National Computer Conference 1979 (AFIPS)*, Vol. 48, pages 313-317, 1979.
8. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *Proc. of EUROCRYPT 2000*, LNCS 1807, pages 139-155. Springer-Verlag, 2000.
9. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proc. of ACM CCS '93*, pages 62-73, 1993.
10. CERT Coordination Center, <http://www.cert.org/>.
11. R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. In *Proc. of EUROCRYPT 2003*, LNCS 2656, pages 255-271, 2003.
12. W. Diffie and M. Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, Vol. IT-22(6), pages 644-654, 1976.
13. Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-Insulated Public Key Cryptosystems. In *Proc. of EUROCRYPT 2002*, LNCS 2332, pages 65-82. Springer-Verlag, 2002.
14. A. Frier, P. Karlton, and P. Kocher. The SSL 3.0 Protocol. Netscape Communications Corp., 1996, <http://wp.netscape.com/eng/ss13/>.
15. O. Goldreich and Y. Lindell. Session-Key Generation using Human Passwords Only. In *Proc. of CRYPTO 2001*, LNCS 2139, pages 408-432, 2001.
16. R. Gennaro and Y. Lindell. A Framework for Password-based Authenticated Key Exchange. In *Proc. of EUROCRYPT 2003*, LNCS 2656, pages 524-543. Springer-Verlag, 2003, A full paper is available at <http://eprint.iacr.org/2003/032>.

17. A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Public Key and Signature Systems. In *Proc. of ACM CCS '96*, pages 100-110, April 1997.
18. IEEE Std 1363-2000. IEEE Standard Specifications for Public Key Cryptography. Main Document, pages 53-57, IEEE, August 29, 2000.
19. IEEE P1363.2. Standard Specifications for Password-based Public Key Cryptographic Techniques. Draft version 11, August 12, 2003.
20. IETF (Internet Engineering Task Force). Challenge Handshake Authentication Protocol. <http://www.ietf.org/rfc/rfc1994.txt>.
21. IETF (Internet Engineering Task Force). IP Security Protocol (ipsec) Charter. <http://www.ietf.org/html.charters/ipsec-charter.html>.
22. IETF (Internet Engineering Task Force). Secure Shell (secsh) Charter. <http://www.ietf.org/html.charters/secsh-charter.html>.
23. IETF (Internet Engineering Task Force). Transport Layer Security (tls) Charter. <http://www.ietf.org/html.charters/tls-charter.html>.
24. G. Itkis and L. Reyzin. SIBIR: Signer-Base Intrusion-Resilient Signatures. In *Proc. of CRYPTO 2002*, LNCS 2442, pages 499-514. Springer-Verlag, 2002.
25. H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. IETF RFC 2104, 1997, <http://www.ietf.org/rfc/rfc2104.txt>.
26. K. Kobara and H. Imai. Pretty-Simple Password-Authenticated Key-Exchange under Standard Assumptions. IACR ePrint Archive, 2003, <http://eprint.iacr.org/2003/038>.
27. J. Katz, R. Ostrovsky, and M. Yung. Efficient Password-Authenticated Key Exchange using Human-Memorable Passwords. In *Proc. of EUROCRYPT 2001*, LNCS 2045, pages 475-494. Springer-Verlag, 2001.
28. T. Kwon. Authentication and Key Agreement via Memorable Password. In *Proc. of NDSS 2001 Symposium*, 2001.
29. P. MacKenzie. More Efficient Password-Authenticated Key Exchange. In *Proc. of Topics in Cryptology (CT-RSA 2001)*, LNCS 2020, pages 361-377, 2001.
30. P. MacKenzie. On the Security of the SPEKE Password-Authenticated Key Exchange Protocol. IACR ePrint Archive, 2001, <http://eprint.iacr.org/2001/057/>.
31. Microsoft Corporation, <http://www.microsoft.com/>.
32. P. MacKenzie, S. Patel, and R. Swaminathan. Password-Authenticated Key Exchange Based on RSA. In *Proc. of ASIACRYPT 2000*, LNCS 1976, pages 599-613. Springer-Verlag, 2000.
33. P. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold Password-Authenticated Key Exchange. In *Proc. of CRYPTO 2002*, LNCS 2442, pages 385-400. Springer-Verlag, 2002.
34. M. Naor and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Applications. In *Proc. of STOC '98*, pages 33-43, 1998.
35. R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. In *Proc. of 10th Annual ACM Symposium on Principles of Distributed Computing*, 1991.
36. M. D. Raimondo and R. Gennaro. Provably Secure Threshold Password-Authenticated Key Exchange. In *Proc. of EUROCRYPT 2003*, LNCS 2656, pages 507-523. Springer-Verlag, 2003.
37. A. Shamir. How to Share a Secret. In *Proc. of Communications of the ACM*, Vol. 22(11), pages 612-613, 1979.
38. A full version of this paper will appear in IACR ePrint Archive.

Untraceable Fair Network Payment Protocols with Off-Line TTP

Chih-Hung Wang

Department of Computer Science and Information Engineering
National Chiayi University
300 University Road, Chiayi, Taiwan 600, R.O.C.
wangch@mail.ncyu.edu.tw

Abstract. A fair network payment protocol plays an important role in electronic commerce. The *fairness* concept in payments can be illustrated as that two parties (e.g. customers and merchants) exchange the electronic items (e.g. electronic money and goods) with each other in a fair manner that no one can gain advantage over the other even if there are malicious actions during exchanging process. In the previous works of fair payments, the buyer is usually required to sign a purchase message which can be traced by everyone. The information about where the buyer spent the money and what he purchased would easily be revealed by this way. This paper employs two techniques of *off-line untraceable cash* and *designated confirmer signatures* to construct a new fair payment protocol, in which the untraceability (or privacy) property can be achieved. A *Restrictive Confirmation Signature Scheme* (RCSS) will be introduced and used in our protocol to prevent the interested persons except the off-line TTP (Trusted Third Party) from tracing the buyer's spending behavior.

Keywords: Cryptography, Electronic cash, Payment System, Undeniable Signature, Designated Confirmer Signatures, Electronic Commerce.

1 Introduction

How the two parties, buyer and merchant, exchange the currency and electronic goods through the network in a fair manner is the crux of the problem on electronic transactions. Since most of the electronic businesses are conducted on an open and insecure network, how to prevent the abnormal behavior, such as malicious termination of the payment process, becomes a critical security consideration on designing a fair payment protocol. A buyer who makes a payment in the network is usually worried that the merchant may refuse to deliver the soft goods though he has sent the money. On the other hand, a merchant will worry that he cannot receive the deserved money after the delivery of goods. Since these two parties do not trust each other, no one wants to send his secret data until receiving the other's.

Two approaches to the achievement of fair exchange have been proposed. The first one is that two parties exchange data simultaneously [EGL85, OO94].

A simplified example to provide simultaneity is that they disclose the secret data bit by bit. This kind of scheme has a drawback that it requires many steps of interactions for exchanging data. In addition, one of these two parties will have an advantage of obtaining one more bit if he maliciously aborts in the middle of the protocol. The second approach is that a trusted third party (TTP) is involved in the exchange process. A straightforward method is that an on-line TTP who acts as a mediator receives the data from both parties in each transaction and then forwards them to the accurate receivers [DGLW96,ZG96]. However, TTP would become a bottleneck on communications since he takes part in all transactions, including the normal cases in which two parties honestly deliver their data. To improve the performance, a novel model called the off-line TTP has been proposed. In this model, TTP is required to participate in the exchange protocols only when the abnormal terminations or faults occur [ASW00,ZG97,BDM98,BF98,Che98]. That means TTP is always able to solve the disputes between two parties but he need not take part in all transactions.

Previously, fair payments seemed to be achieved by use of fair exchange on signatures. For example, two parties can exchange the secret message (soft goods) and the signatures on purchase information. In [ASW98,BDM98,ASW00], a general concept of the fair exchange on signature with off-line TTP is explicated as that one party A sends the encrypted signature to B and convinces B that it is valid and can be decrypted by TTP, without revealing the content of signature. If B completes the verification, he will send his signature (or secret data) to A . In a normal case, A should send his correct signature to B after he received B 's signature. However, if A maliciously aborts the protocol and refuses to send B his signature, B can deliver A 's encrypted signature to the off-line TTP for decryption. The main technique used in these papers is called *verifiable encryption protocol* or *escrow system* [Sta96,Mao97]. However, a generic and efficient construction on verifiable encryption is difficult to implement. Bao et al. [BDM98] in their paper proposed a special implementation with the modified GQ signature algorithm, in which they claimed that the verifiable encryption protocol of the scheme was quite efficient. Unfortunately, Boyd et al. showed that the fairness could be destroyed because the receiver (or any observer) could directly calculate the sender's signature from the encrypted signature without the help of TTP [BF98].

Recently, Boyd et al. [BF98] and Chen [Che98] proposed the efficient fair exchange protocols with off-line TTP by using *verifiable confirmation signatures* (Boyd et al. called them designated converter signatures to emphasize their conversion property). A designated third party, e.g. TTP, can verify the original signatures with interactive protocol or convert the signatures into the self-authenticated signatures which can be verified by everyone. Their proposed schemes are generic constructions for fair exchange and can efficiently run over the Internet.

Our Contributions. Pervious works of fair exchange are not really suitable for many applications on network payments because they are only used to exchange the confidential data or signatures. Especially, many payment applications need

to protect the buyer's purchase privacy, which has never been considered in the previous papers. In our view, a complete solution for fair payment should contain payment actions, such as electronic cash or network credit card method, instead of simply signing the purchase information. Our proposed protocol is the first work to provide a protection on buyer's privacy and it can be regarded as a process of fairly exchanging electronic coins and secret information. The main contributions in this paper are listed as follows:

1. Propose a generic model for *real* fair network payments.
2. Apply a subtle tool of *Restrictive Confirmation Signature Scheme* (RCSS) to achieve the property of untraceability.
3. Design a new technique of *pseudo e-coin* to achieve fairness of exchanging the electronic cash.
4. Demonstrate how to construct a practical and efficient fair network payment protocol based on the Brands' e-cash scheme [Bra93b].

The rest of the paper is organized as follows. We describe the basic model of untraceable fair payment protocol in Section 2. In Section 3, we introduce an useful scheme called Restrictive Confirmation Signature Scheme (RCSS), a basic component for establishing our new protocol. In Section 4, we combines the RCSS and the Brands' electronic cash scheme to realize our protocol. In Section 5, we show the security analysis and properties discussion. Finally, the concluding remarks and future researches are given in Section 6.

2 The Basic Model

We abstractly describe our works in this section. Assume that four parties: the buyer (\mathcal{U}), the merchant (\mathcal{M}), the bank (\mathcal{B}) and the trusted third party (TTP) are involved in the protocol. In a general e-cash scheme, *fairness* can not be achieved because the buyer is required to send *true* electronic coins (e-coins) to the merchant. Instead, this paper designs a technique of *pseudo e-coin* which can be converted to a true one by TTP. The buyer applies the Restrictive Confirmation Signature Scheme (RCSS) (described in Section 3) to sign an order agreement that contains the buyer's and the merchant's names, price of goods, purchase date/information and some other parameters. The RCSS can properly protect the buyer's purchase information by restricting the confirmer's confirmation capability on the signature.

Definition 1. (Restrictive Confirmation Signature Scheme (RCSS)). Let $Sign_{DCS}(S, C, m)$, which is signed by S and can be confirmed by C , be a designated confirmer signature [Cha94] (or called a confirmation signature by [Che98]) on the message m . Assume that a group of verifiers $\mathcal{G} = \{V_i\}_{i=1, \dots, n}$ are pre-determined by S . We say that $Sign_{RCSS}(S, C, \mathcal{G}, m)$ is a restrictive confirmation signature on m if C can convince only some specified verifiers $V_i \in \mathcal{G}$ that $Sign_{RCSS}(S, C, \mathcal{G}, m)$ is valid and truly signed by S .

Three procedures similar to a general e-cash (withdrawal, payment and deposit) are briefly depicted in the following. When a dispute occurs, the TTP is required to participate in an additional procedure *Disputes* to force the completion of the payment process.

Withdrawal. The buyer \mathcal{U} withdraws the money from the bank \mathcal{B} . A blind signature applied here can guarantee the unlinkability for the bank. The withdrawal procedure in our protocol is the same as the one in the general e-cash scheme. After this procedure, \mathcal{U} obtains an electronic coin which can be directly paid to the merchant.

Payment. The buyer \mathcal{U} and the merchant \mathcal{M} exchange the electronic money and goods in this procedure. We assume \mathcal{U} and \mathcal{M} negotiate an order agreement that contains merchandise items and price. The buyer \mathcal{U} then sends enough pseudo e-coins and a signature of RCSS on the order agreement to \mathcal{M} . To prevent the merchant from maliciously delivering the flawed goods, the buyer doesn't send true e-coins to the merchant until he checks and accepts the goods.

1. The buyer \mathcal{U} selects the goods from merchant \mathcal{M} 's web and signs an order agreement:

$$\theta = \text{Sign}_{RCSS}(\mathcal{U}, \mathcal{M}, TTP, OA),$$

where $OA = \{ID_{\mathcal{U}}, ID_{\mathcal{M}}, \text{purchase date/information, goods description, coin parameters}\}$.

2. The buyer \mathcal{U} sends the pseudo e-coins and θ for the goods to the merchant \mathcal{M} .
3. The merchant \mathcal{M} verifies whether the pseudo e-coins and θ are valid. If both checks pass, \mathcal{M} sends the goods to \mathcal{U} . The merchant \mathcal{M} can gain a conviction in this step that he can prove the validity of θ to TTP and ask TTP convert the pseudo e-coins into true e-coins if some faults occur in the rest of payment process.
4. \mathcal{U} checks the goods delivered by \mathcal{M} . If the goods is valid, \mathcal{U} sends his true e-coins to \mathcal{M} .

Disputes. Two possible disputes may occur during payment. \mathcal{M} may refuse to send \mathcal{U} the goods or cheat \mathcal{U} by sending flawed goods. In this case, \mathcal{U} will not send the true e-coins to \mathcal{M} if he does not receive or accept the goods. On the other hand, \mathcal{U} may refuse to send the true e-coins to \mathcal{M} after he receives the valid goods. If so, \mathcal{M} will begin the following procedure to ask TTP convert the pseudo e-coins into true ones.

1. The merchant \mathcal{M} sends pseudo e-coins, OA and θ to TTP and proves that θ is a valid signature and truly signed by \mathcal{U} . Note that no one except \mathcal{M} and TTP can be convinced that θ is valid, since RCSS is applied to the construction of θ .
2. \mathcal{M} privately sends goods to TTP. TTP checks whether the specification of the goods is consistent with the field of goods description written on OA . If yes, TTP sends \mathcal{M} a transformation certificate ($TCer$) which can be used for the conversion of the pseudo e-coins.

An abnormal action is addressed here that \mathcal{M} may abort the step 3 in the payment procedure and directly ask TTP to send him $TCer$ after he receives the pseudo e-coins. However, \mathcal{M} must send TTP the valid goods since TTP has the responsibility to carefully check the goods specification.

Deposit. Generally, the merchant \mathcal{M} can forward the payment transcript, including the true e-coins, to the bank. However, if the payment process is maliciously aborted by \mathcal{U} , \mathcal{M} can send the partial payment transcript with pseudo e-coins plus the transformation certificate ($TCer$) delivered by TTP to the bank for deposit.

3 The Restrictive Confirmation Signature Scheme

In the general designated confirmer signature [Cha94,Oka94,MS98,NMV99], a confirmer can help *every* recipient prove the validity of the signature to others. That means the confirmer has the complete capability of deciding who will benefit from being convinced by a signature. However, this property doesn't meet the requirements of our protocol. In this section, we will illustrate how to construct a Restrictive Confirmation Signature Scheme (RCSS). The basic structure of RCSS is similar to [WC03] but both schemes have different purposes. The concept of RCSS is that we disallow that the confirmer arbitrarily chooses the verifiers; the signer predetermines one or more verifiers whom the confirmer can convince later. We provide a nice approach to add the simulatability into an undeniable signature [CA89,Cha90,CHP92,GKR97]. Hence the signer can later create the proofs in a non-interactive way to delegate confirmer the capability of confirmation of the signature.

In the following, we first give some informal definitions and techniques used in this scheme.

Definition 2. (Trap-Door Commitment (also see [BCC88,JSI96])).

Let c be a function with input (y, u, v) . The notation y denotes the public key of the user whose corresponding secret key is x , u is a value committed to and v is a random number. We say c is a trap-door commitment if and only if it satisfies the following requirements:

1. No polynomial algorithm, when given y , can find two different pairs of (u_1, v_1) and (u_2, v_2) such that $c(y, u_1, v_1) = c(y, u_2, v_2)$.
2. No polynomial algorithm, when given y and $c(y, u, v)$, can find u .
3. There exists a polynomial algorithm that, when given the secret x , (u_1, v_1) and a randomly selected number u_2 , can find v_2 such that $c(y, u_1, v_1) = c(y, u_2, v_2)$ (That means the user who knows the secret x , given (u_1, v_1) , can easily forge the committed value by changing u_1 into u_2).

The following example was suggested by [BCC88,JSI96].

Trap-Door Commitment Example

Let p and q be two large primes and $q|p-1$. The notation g denotes a generator of the subgroup, G_q , of Z_p^* of prime order q . The recipient's secret key is $x \in Z_q^*$

and the corresponding public key is $y = g^x \bmod p$. The sender randomly selects $v \in Z_q^*$ and commits the value $u \in Z_q$ into c as the following:

$$c = g^u y^v \bmod p.$$

The sender sends (u, v) to the recipient for decommitting.

Trap-Door Commitment for Multiple Recipients

Jakobsson et al. [JSI96] proposed an efficient trap-door commitment scheme for multiple recipients $P_i, i = 1, 2, \dots, n$. They modified the commitment to be $c = g^u (\prod_{i=1}^n y_i)^v \bmod p$, where y_i denotes P_i 's public key. Each P_i would be convinced by the proof that u cannot be forged by others as long as he knows his secret key has not been compromised. Any other user would not gain this conviction since all $P_i, i = 1, \dots, n$ can collude to cheat him.

Definition 3. (Message-dependent Proof of Equality of the Discrete Logarithm [Pet97]). A message-dependent proof of equality of the discrete logarithm of y_1 to the base g_1 and y_2 to the base g_2 is a two-tuple $(w, z) = \text{Proof}_{\text{LogEQ}}(m, g_1, y_1, g_2, y_2)$, where $w = F(m || g_1 || y_1 || g_2 || y_2 || g_1^z y_1^w || g_2^z y_2^w)$ and F is a collision resistant hash function.

This proof shows that the prover knows the discrete logarithm $x : \log_{g_1}(y_1) \equiv \log_{g_2}(y_2)$. To construct this proof, the prover randomly selects $k \in Z_q^*$ and calculates $w = F(m || g_1 || y_1 || g_2 || y_2 || g_1^k || g_2^k)$ and $z = k - xw \bmod q$.

Definition 4. (Designated Verifier Message-dependent Proof of Equality of the Discrete Logarithm). Let V denote a designated verifier who has a secret key/public key pair $(x_V, y_V = g^{x_V} \bmod p)$. A designated verifier message-dependent proof of equality of the discrete logarithm of y_1 to the base g_1 and y_2 to the base g_2 is a four-tuple $(w, z, u, v) = \text{Proof}_{\text{DVLogEQ}}(m, c, g_1, y_1, g_2, y_2, y_V)$, where $w = F(m || c || g_1 || y_1 || g_2 || y_2 || g_1^z y_1^{(w+u)} || g_2^z y_2^{(w+u)})$ and $c = g^u y_V^v \bmod p$ is a trap-door commitment.

The prover, using this proof, only can convince the designated verifier V that he knows the discrete logarithm $x : \log_{g_1}(y_1) \equiv \log_{g_2}(y_2)$. To construct this proof, the prover randomly selects $u, v, k \in Z_q^*$ and calculates $c = g^u y_V^v \bmod p$, $w = F(m || c || g_1 || y_1 || g_2 || y_2 || g_1^k || g_2^k)$ and $z = k - x(w + u) \bmod q$.

Definition 5. (Interactive Bi-proof of Equality (see [FOO92, MS98])). Fujioka et. al. in 1992 proposed an interactive bi-proof system that either proved $\log_\alpha(Y) = \log_\beta(Z)$ or proved $\log_\alpha(Y) \neq \log_\beta(Z)$. This proof system can be used to construct RCSS in which the confirmer can prove the validity of the signature to the pre-determined verifiers. We use $\text{BP}(\alpha, Y, \beta, Z)$ to represent this proof system. We omit the detail protocol here, the reader can refer to [FOO92].

Construction of RCSS

The previous works of designated confirmer signatures used the general self-authenticated signature (e.g. RSA, Schnorr [Sch91] and extended Fiat-Shamir

scheme) to construct their schemes. However, it is difficult for these schemes to restrict the confirmer's confirmation capability. Here, we use the *message-dependent proof of equality* (in Definition 3 and Definition 4) and *non-interactive undeniable signature* [JSI96] to construct the RCSS. We also use $a = g^t \bmod p$ and $b = y_C^t \bmod p$, where y_C denotes the confirmer's public key, to add the simulatability to the signature. In addition, we slightly modify the *hinging method* described in the scheme of [Cha94] and [MS98]. The following procedure demonstrates how to pre-determine a single verifier for a signer; however, it is easy to construct an extended scheme to multiple verifiers.

- **System Setup.** The parameters p , q and g are the same ones described previously, and F_1 , F_2 are two collision resistant hash functions. The secret key/public key pairs of the signer S , the confirmer C , the recipient R and the verifier V are $(x_S, y_S = g^{x_S} \bmod p)$, $(x_C, y_C = g^{x_C} \bmod p)$, $(x_R, y_R = g^{x_R} \bmod p)$ and $(x_V, y_V = g^{x_V} \bmod p)$, respectively.
- **Signing Protocol.** Assume the signer has signed a undeniable signature (a, b, δ) on message m related to the confirmer's public key, i.e., $a = g^t \bmod p$, $b = y_C^t \bmod p$ and $\delta = (F_1(m||a) + b)^{x_S} \bmod p$ (note that t is randomly selected by S). For delegating C the ability of confirming this signature, the signer randomly selects k, u, v_1, v_2 and constructs a proof of

$$(w, z, u, v_1, v_2) = \text{Proof}_{DV\text{Log}EQ}(c, g, y_S, F_1(m||a) + b, \delta, y_V),$$

where $c = (c_1||c_2)$, $c_1 = g^u y_V^{v_1} \bmod p$, $c_2 = g^u y_C^{v_2} \bmod p$, $w = F_2(c||g||y_S||F_1(m||a) + b||\delta||g^k||(F_1(m||a) + b)^k)$ and $z = k - x_S(w + u) \bmod q$. Note that we eliminate the first parameter m in $\text{Proof}_{DV\text{Log}EQ}$ because the message has been included in other parameters: $F_1(m||a) + b$ and δ . Thus, the RCSS on m denotes $\text{Sign}_{RCSS}(S, C, V, m) = (a, b, u, v_1, v_2, w, z, \delta)$.

- **Proof by the Signer.** In the original definition of designated confirmer signature scheme, the signer can convince the recipient R that a confirmer C can help R prove the validity of the signature to V . However, according to our basic model in Section 2, the confirmer C also plays the role of the recipient R . That means C will be convinced that he is able to prove the validity of the signature to V in this procedure. C checks the proof by computing $c = ((g^u y_V^{v_1} \bmod p) || (g^u y_C^{v_2} \bmod p))$ and verifying

$$w \stackrel{?}{=} F_2(c||g||y_S||F_1(m||a) + b||\delta||g^z y_S^{(w+u)} || (F_1(m||a) + b)^z \delta^{(w+u)}).$$

To prove the relation of a and b , the signer needs to run the interactive protocol of bi-proof $BP(g, a, y_C, b)$ (see Definition 5) to show $\log_g(a) \equiv \log_{y_C}(b)$.

- **Confirmation Protocol.** The confirmer C can prove the validity of the signature to V by running the interactive protocol bi-proof $BP(g, y_C, a, b)$ with V to show $\log_g(y_C) \equiv \log_a(b)$. The verifier V needs to check whether the signature $(a, b, u, v_1, v_2, w, z, \delta)$ is created properly, and he can be convinced that the signature is valid if he accepts the proof of $BP(g, y_C, a, b)$.

- **Conversion Protocol.** The confirmer can convert the designated confirmer signature to a general non-interactive undeniable signature. Since the signer has constructed the designated verifier proof in a non-interactive way, V can check the validity of the signature by himself. The verifier V no longer needs to ask C to help him verify the signature. Here, C randomly selects $\sigma \in Z_q^*$ and computes $E = a^\sigma \bmod p$ and $T = \sigma + x_C F(a, E) \bmod q$, where F is also a hash function. The confirmer sends (E, T) to the verifier V , thus, V can verify $a^T \stackrel{?}{=} Eb^{F(a, E)}$ [Cha94].

Security of RCSS

Here, some security properties will be considered for RCSS.

Unforgeability. The forgeability problems that the intruder I tries to forge (a^*, b^*, δ^*) without access to secret key x_S , can be illustrated with two scenarios. The first one is that I selects a message m^* , a^* and computes $b^* = F_1(m||a) + b - F_1(m^*||a^*)$. However, the b^* which I can easily calculate would not have the same discrete logarithm as a^* has because F_1 is a collision resistant hash function whose output is approximately random. The second one is that I randomly selects $t^* \in Z_q^*$ and compute $a^* = g^{t^*}$ and $b^* = y_C^{t^*}$. In this attack scenario, I can not find a proper m^* to satisfy the equation $F_1(m^*||a^*) + b^* = F_1(m||a) + b$ since inverting an one-way hash function F_1 , given its output, is computationally infeasible.

Indistinguishability. Given a random number a^* , a simulated signature on the message m^* can be represented as $(a^*, b^*, u, v1, v2, w, z, \delta)$ where $b^* = F_1(m||a) + b - F_1(m^*||a^*)$. The verifier cannot distinguish between the correct signature and simulated signature because he knows nothing about the discrete logarithm of a^* to the base g and b^* to the base y_C . Hence, without confirmer's help, the verifier would not be convinced that both discrete logarithms of a^* and b^* are equal. The indistinguishability of RCSS can also be proved by Decision-Diffie-Hellman assumption [MS98].

The following lemma shows that no one except the confirmer C and the designated verifier V can be convinced that the RCSS is correctly constructed and truly signed by S . Note that C and V can be convinced by the proof of the signature because they know their secret keys have not been compromised; however, others cannot obtain this conviction since they know that C and V are able to collude to create a simulated transcript to cheat them.

Lemma 1. (Simulating Transcripts of RCSS). *The confirmer C and designated verifier V can collude to create a simulated transcript of RCSS without accessing the signer's secret key x_S . Assume that V randomly selects α_1 and computes $c_1 = g^{\alpha_1} \bmod p$, and C randomly selects α_2 and computes $c_2 = g^{\alpha_2} \bmod p$. Thus they can compute the following simulated transcript by cooperatively choosing the random numbers $\beta, \tau, z \in Z_q^*$:*

$$\begin{aligned} c &= (c_1||c_2), \\ a &= g^\tau \bmod p, \end{aligned}$$

$$\begin{aligned}
 b &= y_C^\tau \bmod p, \\
 w &= F_2(c||g||y_S||F_1(m^*||a) + b||\delta^*||g^z y_S^\beta||(F_1(m^*||a) + b)^z \delta^{*\beta}), \\
 u &= (\beta - w) \bmod q.
 \end{aligned}$$

V and C individually computes v_1 and v_2 as below:

$$\begin{aligned}
 v_1 &= (\alpha_1 - u)(x_V)^{-1} \bmod q, \\
 v_2 &= (\alpha_2 - u)(x_C)^{-1} \bmod q.
 \end{aligned}$$

4 The Realization of Our Fair Network Payment Model

Brands in 1993 proposed a nice approach to untraceable electronic cash [Bra93b]. In this section, we will present an untraceable fair payment protocol based on a modification of Brands scheme. We develop a pseudo e-coin technique combined into the payment procedure. Some mathematic definitions are omitted here, and the reader can refer [Bra93b] for further details.

The concept of pseudo e-coin technique is to create a designated confirmer signature (DCS) by which the merchant can be convinced that there exists a trusted third party (TTP) who can convert DCS into a self-authenticated signature. Therefore, if the merchant later does not receive the true e-coins from the buyer, he would ask TTP for a transformation certificate $TCer$.

We explicate an off-line fair payment in the following procedures. For simplifying the notation, we redefine all symbols in this section except some common parameters such as p , q and g (Note that the symbols used in this section have different definitions from that in Section 3).

Setup. Let p and q be two large primes as defined in Section 3. The bank \mathcal{B} publishes a generator-tuple (g, g_1, g_2) in G_q and two collision-resistant hash functions $\mathcal{H} : G_q \times G_q \times G_q \times G_q \times G_q \times G_q \rightarrow Z_q^*$ and $\mathcal{H}_0 : G_q \times G_q \times ID \times DATE/TIME \rightarrow Z_q^*$. \mathcal{B} also generates a random number $x_{\mathcal{B}} \in Z_q^*$ as his secret key corresponding to a public key $y_{\mathcal{B}} = g^{x_{\mathcal{B}}} \bmod p$.

Account Opening. The buyer \mathcal{U} randomly selects $u_1 \in Z_q^*$ and transmits $I = g_1^{u_1} \bmod p$ to \mathcal{B} if $I g_2 \neq 1$. The identifier I used to uniquely identify \mathcal{U} can be regarded as the account number of \mathcal{U} . Then \mathcal{B} publishes $g_1^{x_{\mathcal{B}}} \bmod p$ and $g_2^{x_{\mathcal{B}}} \bmod p$ so that \mathcal{U} can compute $z = (I g_2)^{x_{\mathcal{B}}} = (g_1^{x_{\mathcal{B}}})^{u_1} g_2^{x_{\mathcal{B}}} \bmod p$ for himself¹.

Withdrawal. The buyer \mathcal{U} performs the following protocol to withdraw a single e-coin from the bank:

¹ Chan et al. [CFMT96] have proposed a problem of mis-representation of identities for Brands' scheme (Brands commented that it is only an inadvertent omission and the similar result has been presented in [Bra93a]). This problem can be efficiently solved by applying a minimal-knowledge proof to prove the correct construction of I during the account opening stage.

1. \mathcal{B} randomly selects $w \in Z_q^*$ and sends $e_1 = g^w \bmod p$ and $e_2 = (Ig_2)^w \bmod p$ to \mathcal{U} .
2. \mathcal{U} randomly selects s, x_1 and x_2 in Z_q^* and computes $A = (Ig_2)^s \bmod p$, $B = g_1^{x_1} g_2^{x_2} \bmod p$ and $z' = z^s \bmod p$. \mathcal{U} also randomly selects u, v and t_c in Z_q^* and computes $e_1' = e_1^u g^v \bmod p$, $e_2' = e_2^{su} A^v \bmod p$ and $(a_c, b_c) = (g^{t_c} \bmod p, y_{TTP}^{t_c} \bmod p)$. Then \mathcal{U} sends $c = c'/u \bmod q$ to \mathcal{B} , where $c' = \mathcal{H}(A, B, z', e_1', e_2', b_c) + a_c \bmod q$. Note that (a_c, b_c) is a pair of confirmation parameters.
3. \mathcal{B} sends $r = cx_{\mathcal{B}} + w \bmod q$ to \mathcal{U} .
4. \mathcal{U} verifies whether $g^r = y_{\mathcal{B}}^c e_1 \bmod p$ and $(Ig_2)^r = z^c e_2 \bmod p$. If the verification holds, \mathcal{U} accepts and computes $r' = ru + v \bmod q$. Note that $\langle A, B, (z', e_1', e_2', r', a_c, b_c) \rangle$ represents a single pseudo e-coin.

Payment. The buyer \mathcal{U} and the merchant \mathcal{M} exchange the e-coins and the soft goods in this procedure. The following protocol will be done (Note that we add the subscripts to some symbols to represent the multiple e-coins).

1. The buyer \mathcal{U} selects goods and signs an order agreements

$$\theta = \text{Sign}_{RCSS}(\mathcal{U}, \mathcal{M}, TTP, OA),$$

where $OA = \{ID_{\mathcal{U}}, ID_{\mathcal{M}}, \text{purchase date/information, goods description}, (A_i, B_i)_{i=1,2,\dots,n}\}$ and n denotes the number of e-coins for the goods which \mathcal{U} wants to buy.

2. The buyer \mathcal{U} sends the unused e-coins $\langle A_i, B_i, (z'_i, e'_{1i}, e'_{2i}, r'_i, a_{ci}, b_{ci}) \rangle$, for $i = 1, 2, \dots, n$, to \mathcal{M} .
3. The merchant \mathcal{M} verifies the pseudo e-coins and θ . If all of them are valid and $A_i \neq 1$, for $i = 1, 2, \dots, n$, then he sends $d_i = \mathcal{H}_0(A_i, B_i, ID_{\mathcal{M}}, \text{date/time})$ to \mathcal{U} . et al.
4. The buyer \mathcal{U} sends $k_{1i} = d_i(u_{1i}s_i) + x_{1i} \bmod q$ and $k_{2i} = d_i s_i + x_{2i} \bmod q$, for $i = 1, 2, \dots, n$, to the merchant \mathcal{M} . In addition, the buyer \mathcal{U} must run the interactive protocol of bi-proof $BP(g, a_{ci}, y_{TTP}, b_{ci})$ with \mathcal{M} to show all $\log_g(a_{ci}) \equiv \log_{y_{TTP}}(b_{ci})$.
5. The merchant \mathcal{M} will accept these pseudo e-coins and payment transcripts $\langle A_i, B_i, (z'_i, e'_{1i}, e'_{2i}, r'_i, a_{ci}, b_{ci}), (d_i, k_{1i}, k_{2i}) \rangle$, for $i = 1, 2, \dots, n$, if the following verifications hold:

$$\begin{aligned} g^{r'_i} &= y_{\mathcal{B}}^{\mathcal{H}(A_i, B_i, z'_i, e'_{1i}, e'_{2i}, b_{ci}) + a_{ci}} e'_{1i}, \\ A_i^{r'_i} &= z_i^{\mathcal{H}(A_i, B_i, z'_i, e'_{1i}, e'_{2i}, b_{ci}) + a_{ci}} e'_{2i}, \text{ and} \\ g_1^{k_{1i}} g_2^{k_{2i}} &= A_i^{d_i} B_i. \end{aligned}$$

If the above verifications pass, the merchant \mathcal{M} sends the soft goods to the buyer \mathcal{U} .

6. The buyer \mathcal{U} checks the soft goods delivered by \mathcal{M} . If it is flawless, he releases t_{ci} , for $i = 1, 2, \dots, n$, to the merchant \mathcal{M} . Since each one can check $a_{ci} = g^{t_{ci}} \bmod p$ and $b_{ci} = y_{TTP}^{t_{ci}} \bmod p$ by himself, the coin $\langle A_i, B_i, (z'_i, e'_{1i}, e'_{2i}, r'_i, a_{ci}, b_{ci}, t_{ci}), (d_i, k_{1i}, k_{2i}) \rangle$ denotes a *true* e-coin that can be directly cashed from the bank.

Disputes. If \mathcal{U} refuses to send t_{ci} to the merchant \mathcal{M} (see the Step 6 in the Payment procedure), \mathcal{M} will begin the dispute process in which the TTP can convert the pseudo e-coins into the true e-coins.

1. The merchant \mathcal{M} sends the order agreement OA , the signature θ , soft goods and pseudo e-coins $\langle A_i, B_i, (z'_i, e'_{1i}, e'_{2i}, r'_i, a_{ci}, b_{ci}), (d_i, k_{1i}, k_{2i}) \rangle$, for $i = 1, 2, \dots, n$, to TTP.
2. The TTP checks the validity of the soft goods, pseudo e-coins and signature θ . If the pseudo e-coins are constructed properly, the soft goods transmitted from \mathcal{M} is consistent with the description in OA , and θ is valid, TTP sends \mathcal{M} a transformation certificate $TCer = (E_{ci}, T_{ci})$, for $i = 1, 2, \dots, n$, to \mathcal{M} , where $E_{ci} = a_{ci}^{\sigma_i} \bmod p$ (σ_i is a random number selected by TTP) and $T_{ci} = \sigma_i + x_{TTP} F(a_{ci}, E_{ci}) \bmod q$. The transformation certificate can be used to verify the relation of a_{ci} and b_{ci} by the following equation:

$$a_{ci}^{T_{ci}} \stackrel{?}{=} E_{ci} b_{ci}^{F(a_{ci}, E_{ci})} \bmod p$$

3. TTP sends the soft goods to the buyer \mathcal{U} .

Deposit. In a normal case, \mathcal{M} forwards the payment transcript and the true e-coins $\langle A_i, B_i, (z'_i, e'_{1i}, e'_{2i}, r'_i, a_{ci}, b_{ci}, t_{ci}), (d_i, k_{1i}, k_{2i}) \rangle$, for $i = 1, 2, \dots, n$, to the bank for deposit. Nevertheless, if the buyer \mathcal{U} maliciously aborts the payment process, \mathcal{M} can start the dispute process to acquire the $TCer$ from TTP. In this situation, the pseudo e-coins $\langle A_i, B_i, (z'_i, e'_{1i}, e'_{2i}, r'_i, a_{ci}, b_{ci}), (d_i, k_{1i}, k_{2i}) \rangle$ plus $TCer = (E_{ci}, T_{ci})$, for $i = 1, 2, \dots, n$, can be the valid tokens for deposit. We also can regard $\langle A_i, B_i, (z'_i, e'_{1i}, e'_{2i}, r'_i, a_{ci}, b_{ci}), (d_i, k_{1i}, k_{2i}), (E_{ci}, T_{ci}) \rangle$ as a true e-coin with different form.

5 Security Issues

The security of our new protocol relies on Brands' e-cash scheme and RCSS. The following properties are provided to prove the *fairness* and *untraceability* which are both pivotal features in our protocol.

Proposition 1. (Unforgeability). *No one except \mathcal{U} can create his own pseudo e-coins $\langle A_i, B_i, (z'_i, e'_{1i}, e'_{2i}, r'_i, a_{ci}, b_{ci}), (d_i, k_{1i}, k_{2i}) \rangle$, for $i = 1, 2, \dots, n$.*

This proposition holds because the Brand's e-cash scheme is secure. The possible scenario of forging the e-coins is that the attacker randomly selects $u_{1i}, \bar{s}_i, x_{1i}$ and x_{2i} in Z_q^* and computes $\bar{A}_i = (g_1^{u_{1i}} g_2)^{\bar{s}_i} \bmod p$ and $\bar{B}_i = g_1^{x_{1i}} g_2^{x_{2i}} \bmod p$. In this case, the attacker can randomly select \bar{z}'_i, \bar{r}'_i and λ_i to compute $\bar{e}'_{1i} = g^{\bar{r}'_i} y_B^{-\lambda_i}$ and $\bar{e}'_{2i} = \bar{A}_i \bar{r}'_i \bar{z}'_i^{-\lambda_i}$. The purpose of the attacker is to find the proper a_{ci}^* and b_{ci}^* such that $\lambda_i = \mathcal{H}(\bar{A}_i, \bar{B}_i, \bar{z}'_i, \bar{e}'_{1i}, \bar{e}'_{2i}, b_{ci}^*) + a_{ci}^*$. However, though the attacker can easily calculate $a_{ci}^* = \lambda_i - \mathcal{H}(\bar{A}_i, \bar{B}_i, \bar{z}'_i, \bar{e}'_{1i}, \bar{e}'_{2i}, b_{ci}^*)$ by randomly selecting a value of b_{ci}^* , it is computationally infeasible for the attacker to find a_{ci}^* and b_{ci}^* which have the same discrete logarithm because \mathcal{H} is a collision resistant hash function whose output is approximately random.

Proposition 2. (Indistinguishability). *No one can distinguish between a valid pseudo e-coin and a simulated one without the help of the buyer or TTP.*

According to Proposition 1, a simulated pseudo e-coin can be represented as $\langle \bar{A}_i, \bar{B}_i, (\bar{z}'_i, \bar{e}'_{1i}, \bar{e}'_{2i}, \bar{r}'_i, a_{ci}^*, b_{ci}^*), (d_i, k_{1i} = d_i(\bar{u}_{1i}\bar{s}_i) + \bar{x}_{1i}, k_{2i} = d_i\bar{s}_i + \bar{x}_{2i}) \rangle$. Any interested party, such as a bank, cannot distinguish between a properly constructed pseudo e-coin and a simulated pseudo e-coin without the help of the buyer or TTP, because the bank knows nothing about the discrete logarithm of a_{ci}^* to the base g and b_{ci}^* to the base y_{TTP} . That means the bank cannot be convinced that the discrete logarithms of both a_{ci}^* and b_{ci}^* are equal. This property indicates the fairness that even if the buyer \mathcal{U} sent the pseudo e-coins to the merchant \mathcal{M} before he receives the soft goods, the merchant \mathcal{M} cannot gain the advantage over \mathcal{U} .

Proposition 3. (Convertibility). *If \mathcal{M} accepts the pseudo e-coins, it is guaranteed that TTP can later convert the pseudo e-coins into the true e-coins which can be directly deposited in the bank.*

This proposition can be proven by the confirmation signatures [Cha94, MS98]. The merchant \mathcal{M} cannot accept an invalid pseudo e-coin except with negligible probability.

Lemma 2. (Fairness). *If the propositions of unforgeability, indistinguishability, and convertibility hold for our newly proposed payment protocol, it can be guaranteed that, at the end of the transaction, the buyer \mathcal{U} can obtain the soft goods if and only if the merchant \mathcal{M} can gain the equivalent true e-coins.*

Clearly, if two parties of \mathcal{U} and \mathcal{M} are honest, the fairness can be achieved without interacting with TTP. The rest of the condition is that one of \mathcal{U} and \mathcal{M} is dishonest. The unforgeability can guarantee that \mathcal{U} cannot fool \mathcal{M} by delivering the invalid pseudo e-coins, and the convertibility can prevent \mathcal{U} from refusing to send true e-coins or sending the forged e-coins to \mathcal{M} . On the other hand, if \mathcal{M} is dishonest, he may refuse to send valid goods to \mathcal{U} after he receives the valid pseudo e-coins. However, because of the indistinguishability, \mathcal{M} cannot receive the useful e-coins for deposit if he cheats during the payment procedure.

Lemma 3. (Untraceability). *No one except \mathcal{M} and TTP can confirm the signature θ . That means only \mathcal{M} and TTP can be convinced that the order agreement OA is valid.*

This lemma holds because the signature θ is created by RCSS. Thus \mathcal{M} can only convince TTP that θ is really signed by \mathcal{U} . The security of RCSS has been discussed in Section 3.

Lemma 4. (Unlinkability). *The bank or other parties can not link a coin $\langle A_i, B_i, (z'_i, e_{1i}', e_{2i}', r'_i, a_{ci}, b_{ci}) \rangle$ to the original owner.*

This lemma can be proven by using blind signature property of withdrawal procedure in [Bra93b].

Coin Size. Compared to the Brands' scheme, the individual coin of our protocol has extra three items: a_c , b_c and t_c . The total size of these items is $2|p| + |q|$. Especially, in the dispute condition, TTP is required to release $TCer$ with the size of $|p| + |q|$.

6 Conclusions

Electronic cash is considered to have a significant advantage over network credit card because the former can properly protect the buyer's payment privacy. In the proposed paper, we have presented a general model in which two parties can fairly exchange the electronic cash and soft goods. Our new scheme is also the first one that can provide the untraceability property on fair payments.

The future research is addressed here that we are planning to design the fair payment protocols with other payment tools, such as the electronic check and the divisible electronic cash. The privacy property, for which we have constructed a generic model, is a critical issue on the design of our future work.

Acknowledgement

This work was supported in part by National Science Council of Republic of China under contract NSC91-2213-E-415-005.

References

- ASW98. N. Asokan, V. Shoup, and M. Waidner. Optimistic Fair Exchange of Digital Signatures. In *Advances in Cryptology - proceedings of Eurocrypt'98*, Lecture Notes in Computer Science (LNCS) 1403, pages 591-606, Springer-Verlag, 1998.
- ASW00. N. Asokan, V. Shoup, and M. Waidner. Optimistic Fair Exchange of Digital Signatures. *IEEE Journal on Selected Areas in Communications*, vol. 18, pages 591-610, Apr. 2000.
- BDM98. F. Bao, R. H. Deng, W. Mao. Efficient and Practical Fair Exchange Protocols with Off-line TTP. *Proceedings of the 1998 IEEE Symposium on Security and Privacy*. IEEE Computer Press, pages 77-85, Oakland, CA, May 1998.
- BF98. C. Boyd and E. Foo. Off-line Fair Payment Protocols Using Convertible Signature. In *Advances in Cryptology - proceedings of Asiacrypt'98*, pages 271-285, Springer-Verlag, 1998.
- Bra93a. S. Brands. An Efficient Off-line Electronic Cash System Based on the Representation Problem. Technical Report CS-R9323, CWI (Centre for Mathematics and Computer Science), Amsterdam, 1993.
<ftp://ftp.cwi.nl/pub/CWIreports/AA/CS-R9323.pdf>.
- Bra93b. S. Brands. Untraceable Off-line Cash in Wallets with Observers. In *Advances in Cryptology - proceedings of Crypto'93*, Lecture Notes in Computer Science (LNCS) 773, pages 302-318, Springer-Verlag, 1993.

- BCC88. G. Brassard, D. Chaum, C. Crepeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences*, Vol. 37, No. 2, pages 156-189, 1988.
- CFMT96. A. Chan, Y. Frankel, P. MacKenzie and Y. Tsionis. Mis-representation of Identities in E-cash Schemes and how to Prevent it. In *Advances in Cryptology - proceedings of Asiacrypt'96*, Lecture Notes in Computer Science (LNCS) 1163, pages 276-285, Springer-Verlag, 1996.
- Cha90. D. Chaum. Zero-knowledge Undeniable Signature. In *Advances in Cryptology - proceedings of Eurocrypt'90*, Lecture Notes in Computer Science (LNCS) 473, pages 458-464, Springer-Verlag, 1990.
- Cha94. D. Chaum. Designated Confirmer Signatures. In *Eurocrypt'94*, pages 86-91, 1994.
- CHP92. D. Chaum, E. van Heijst and B. Pfitzmann. Cryptographically Strong Undeniable Signers, Unconditionally Secure for the Signer. In *Advances in Cryptology - proceedings of Crypto'91*, Lecture Notes in Computer Science (LNCS) 576, pages 470-484, Springer-Verlag, 1992.
- CA89. David Chaum and Hans Van Antwerpen: Undeniable Signature. In *Advances in Cryptology - proceedings of Crypto'89*, Lecture Notes in Computer Science (LNCS) 435, pages 212-217, Springer-Verlag, 1989.
- Che98. L. Chen. Efficient Fair Exchange with Verifiable Confirmation of Signatures. In *Advances in Cryptology - proceedings of Asiacrypt'98*, pages 286-299, Springer-Verlag, 1998.
- DGLW96. R. H. Deng, L. Gong, A. A. Lazar and W. Wang. Practical Protocol for Certified Electronic Mail. *Journal of Network and Systems Management*, vol. 4, no. 3, pages 279-297, 1996.
- EGL85. S. Even, O. Goldreich and A. Lempel. A Randomized Protocol for Signing Contracts. *CACM*, vol. 28, no. 6, pages 637-647, 1985.
- FOO92. A. Fujioka, T. Okamoto, K. Ohta. Interactive Bi-Proof Systems and Undeniable Signature Schemes. In *Advances in Cryptology - proceedings of Eurocrypt'91*, Lecture Notes in Computer Science, pages 243-256, Springer-Verlag, 1992.
- GKR97. R. Gennaro, H. Krawczyk, and T. Rabin. RSA-Based Undeniable Signatures. In *Advances in Cryptology - proceedings of Crypto'97*, Lecture Notes in Computer Science (LNCS) 1294, pages 132-149, Springer-Verlag, 1997.
- GKR99. R. Gennaro, H. Krawczyk and T. Rabin. Undeniable Certificates. *Electronic Letters*, vol. 35, no. 20, pages 1723-1724, Sep. 1999.
- JSI96. M. Jakobsson, K. Sako and R. Impagliazzo. Designated Verifier Proofs and Their Application. In *Advances in Cryptology - proceedings of Eurocrypt'96*, Lecture Notes in Computer Science (LNCS) 1070, pages 143-154, Springer-Verlag, 1996.
- Mao97. W. Mao. Publicly Verifiable Partial Key Escrow. In *ACISP'97*, pages 240-248, Springer-Verlag, 1997.
- MS98. M. Michels and M. Stadler. Generic Constructions for Secure and Efficient Confirmer Signature Schemes. In *Advances in Cryptology - Eurocrypt'98*, Lecture Notes in Computer Science (LNCS) 1403, pages 406-421, Springer-Verlag, 1998.
- NMV99. K. Nguyen, Y. Mu, and V. Varadharajan. Undeniable Confirmer Signature. *Information Security - Proceedings of Second International Workshop, ISW'99*, Lecture Notes in Computer Science (LNCS) 1729, pages 235-246, Springer-Verlag, 1999.

- Oka94. T. Okamoto. Designated Confirmer Signatures and Public-key Encryption Are Equivalent. In *Advances in Cryptology - Crypto'94*, Lecture Notes in Computer Science (LNCS) 839, pages 61-74, Springer-Verlag, 1994.
- OO94. T. Okamoto and K. Ohta. How to Simultaneously Exchange Secrets by General Assumption. *Proceedings of 2nd ACM Conference on Computer and Communications Security*, pages 184-192, 1994.
- Pet97. H. Petersen. How to Convert any Digital Signature Scheme into a Group Signature Scheme, to appear in *Security Protocol'97*, LNCS, Springer Verlag, 1997.
- PS96. D. Pointcheval, J. Stern. Security Proofs for Signature. In *Advances in Cryptology - proceedings of Eurocrypt'96*, Lecture Notes in Computer Science (LNCS) 1070, pages 387-398, Springer-Verlag, 1996.
- Sch91. C. P. Schnorr. Efficient Signature Generation for Smart Cards. *Journal of Cryptology*, vol. 4, no. 3, pages 161-174, 1991.
- Sta96. M. Stadler. Publicly Verifiable Secret Sharing. In *Advances in Cryptology - proceedings of Eurocrypt'96*, Lecture Notes in Computer Science (LNCS) 1070, pages 190-199, Springer-Verlag, 1996.
- WC03. C.-H. Wang and Y.-C Chen. Proxy Confirmation Signatures. *Informatica* (accepted), 2003.
- ZG96. J. Zhou and D. Gollmann. A Fair Non-repudiation Protocol. *Proceedings of the 1996 IEEE Symposium on Security and Privacy*. IEEE Computer Press, pages 55-61, Oakland, CA, 1996.
- ZG97. J. Zhou and D. Gollmann. An Efficient Non-repudiation Protocol. *Proceedings of the 1997 IEEE Computer Security Foundations Workshop (CSFW 10)*. IEEE CS Press, pages 126-132, 1997.

Incremental Multiset Hash Functions and Their Application to Memory Integrity Checking

Dwaine Clarke*, Srinivas Devadas, Marten van Dijk**,
Blaise Gassend, and G. Edward Suh

MIT Computer Science and Artificial Intelligence Laboratory
{declarke,devadas,marten,gassend,suh}@mit.edu

Abstract. We introduce a new cryptographic tool: *multiset* hash functions. Unlike standard hash functions which take strings as input, multiset hash functions operate on multisets (or sets). They map multisets of arbitrary finite size to strings (hashes) of fixed length. They are incremental in that, when new members are added to the multiset, the hash can be updated in time proportional to the change. The functions may be *multiset-collision resistant* in that it is difficult to find two multisets which produce the same hash, or just *set-collision resistant* in that it is difficult to find a set and a multiset which produce the same hash.

We demonstrate how set-collision resistant multiset hash functions make an existing offline memory integrity checker secure against active adversaries. We improve on this checker such that it can use smaller time stamps without increasing the frequency of checks. The improved checker uses multiset-collision resistant multiset hash functions.

Keywords: multiset hash functions, set-collision resistance, multiset-collision resistance, incremental cryptography, memory integrity checking

1 Introduction

Standard hash functions, such as SHA-1 [11] and MD5 [12], map strings of arbitrary finite length to strings (hashes) of a fixed length. They are collision-resistant in that it is difficult to find different input strings which produce the same hash. Incremental hash functions, described in [2], have the additional property that, given changes to the input string, the computation to update the hashes is proportional to the amount of change in the input string. For a small change, incremental hashes can be quickly updated, and do not need to be recalculated over the entire new input.

Multiset hash functions are a novel cryptographic tool, for which the ordering of the inputs is not important. They map multisets of arbitrary finite size to hashes of fixed length. They are incremental in that, when new members are added to the multiset, the hash can be quickly updated. Because multiset

* Note: authors are listed alphabetically.

** Visiting researcher from Philips Research, Prof Holstlaan 4, Eindhoven, The Netherlands.

hash functions work on multisets, we introduce definitions for multiset-collision resistance and set-collision resistance.

In particular, we introduce four multiset hash functions, each with its own advantages. **MSet-XOR-Hash** uses the XOR operation and is very efficient; however, it uses a secret key and is only set-collision resistant. **MSet-Add-Hash** uses addition modulo a large integer and, thus, is slightly less efficient than **MSet-XOR-Hash**; **MSet-Add-Hash** also uses a secret key but it is multiset-collision resistant. **MSet-Mu-Hash** uses finite field arithmetic and is not as efficient as the other two hash functions; however, **MSet-Mu-Hash** is multiset-collision resistant, and unlike the other two hash functions, does not require a secret key. **MSet-VAdd-Hash** is more efficient than **MSet-Mu-Hash**; it is also multiset-collision resistant, and does not use a secret key, but the hashes it produces are significantly longer than the hashes of the other functions.

The proven security of **MSet-XOR-Hash** and **MSet-Add-Hash** is quantitative. We reduce the hardness of finding collisions to the hardness of breaking the underlying pseudorandom functions. The proven security of **MSet-Mu-Hash** is in the random oracle model and is based on the hardness of the discrete logarithm problem. The proven security of **MSet-VAdd-Hash** is also in the random oracle model and is based on the hardness of the worst-case shortest vector problem.

We demonstrate how multiset hash functions enable secure offline integrity checkers for untrusted memory. Checking the integrity of memory is important in building secure processors which can facilitate software licensing and Digital Rights Management (DRM) [13,14].

The paper is organized as follows. Section 2 describes related work and summarizes our contributions. Multiset hash functions are defined in Section 3. **MSet-XOR-Hash** and **MSet-Add-Hash** are described in Section 4; **MSet-Mu-Hash** and **MSet-VAdd-Hash** are described in Section 5. Our application of multiset hash functions to checking the integrity of memory is detailed in Section 6. Section 7 concludes the paper. Appendices A, B, C, and D prove the security of our multiset hash functions. Appendix E proves the security of our memory integrity checker.

2 Related Work and Our Contributions

The main contribution of our work is the introduction of multiset hash functions together with the definition of multiset and set collision resistance. The second contribution is the development of a general theory leading to Theorem 1 from which we derive set-collision resistance for **MSet-XOR-Hash**, a multiset hash based on the XOR operation (addition modulo 2), and multiset-collision resistance for **MSet-Add-Hash**, a multiset hash based on addition modulo a large integer. The theory generalizes the results in [3], where an XOR-based scheme is used for message authentication. Our theory holds for addition modulo any integer.

Both **MSet-XOR-Hash** and **MSet-Add-Hash** use a secret key. The third contribution is Theorem 2 that proves multiset-collision resistance for **MSet-Mu-Hash**, a multiset hash function based on multiplication in a finite field; **MSet-Mu-Hash** does not use a secret key. The proof's basic line of thought is from [4] which

develops message hashing based on multiplication in a finite field. The fourth contribution, leading to **MSet-VAdd-Hash**, is Theorem 3 proving that we may replace multiplication in the finite field by vector addition modulo a large integer. In [4], a similar theorem is used for message hashing. Our theorem (and their theorem) follows directly from application of Ajtai's theorem [1,8].

Our final significant contribution is that we introduce an offline checker that is cryptographically secure against active adversaries, and which improves on the performance of the original offline checker in [6].

3 Multiset Hash Functions

This section describes multiset hash functions. We first introduce multisets. We refer to a multiset as a finite unordered group of elements where an element can occur as a member more than once. All sets are multisets, but a multiset is not a set if an element appears more than once. Let M be a multiset of elements of a countable set B . The number of times $b \in B$ is in the multiset M is denoted by M_b and is called the multiplicity of b in M . The sum of all the multiplicities of M is called the cardinality of M . Multiset union combines two multisets into a multiset in which elements appear with a multiplicity that is the sum of their multiplicities in the initial multisets. We denote multiset union by \cup and assume that the context in which \cup is used makes clear to the reader whether we mean set union or multiset union.

Definition 1. Let $(\mathcal{H}, +_{\mathcal{H}}, \equiv_{\mathcal{H}})$ be a triple of probabilistic polynomial time (ppt) algorithms. That triple is a multiset hash function if it satisfies:

compression: \mathcal{H} maps multisets of B into elements of a set with cardinality $\approx 2^m$, where m is some integer. Compression guarantees that we can store hashes in a small bounded amount of memory.

comparability: Since \mathcal{H} can be a probabilistic algorithm, a multiset need not always hash to the same value. Therefore we need $\equiv_{\mathcal{H}}$ to compare hashes. The following relation must hold for comparison to be possible:

$$\mathcal{H}(M) \equiv_{\mathcal{H}} \mathcal{H}(M)$$

for all multisets M of B .

incrementality: We would like to be able to efficiently compute $\mathcal{H}(M \cup M')$ knowing $\mathcal{H}(M)$ and $\mathcal{H}(M')$. The $+_{\mathcal{H}}$ operator makes that possible:

$$\mathcal{H}(M \cup M') \equiv_{\mathcal{H}} \mathcal{H}(M) +_{\mathcal{H}} \mathcal{H}(M')$$

for all multisets M and M' of B . In particular, knowing only $\mathcal{H}(M)$ and an element $b \in B$, we can easily compute $\mathcal{H}(M \cup \{b\}) = \mathcal{H}(M) +_{\mathcal{H}} \mathcal{H}(\{b\})$.

As it is, this definition is not very useful, because \mathcal{H} could be any constant function. We need to add some kind of collision resistance to have a useful hash function. A collision for M' is a multiset $M \neq M'$ such that $\mathcal{H}(M) \equiv_{\mathcal{H}} \mathcal{H}(M')$. A multiset hash function is *(multi)set-collision resistant* if it is computationally

infeasible to find a (multi)set S of B and a multiset M of B such that the cardinalities of S and M are of polynomial size in m , $S \neq M$, and $\mathcal{H}(S) \equiv_{\mathcal{H}} \mathcal{H}(M)$. The following definition makes this notion formal.

Definition 2. Let a family \mathcal{F} of multiset hash functions $(\mathcal{H}_K, +_{\mathcal{H}_K}, \equiv_{\mathcal{H}_K})$ be indexed by a key (seed) $K \in \mathcal{K}$. For \mathcal{H}_K in \mathcal{F} , we denote by m_K the logarithm of the cardinality of the set into which \mathcal{H}_K maps multisets of B , that is m_K is the number of output bits of \mathcal{H}_K . We define \mathcal{K}_m as the set of keys $K \in \mathcal{K}$ for which $m_K \geq m$. By $\mathcal{A}(\mathcal{H}_K)$ we denote a probabilistic polynomial time (in m_K) algorithm with oracle access to $(\mathcal{H}_K, +_{\mathcal{H}_K}, \equiv_{\mathcal{H}_K})$.

The family \mathcal{F} satisfies (multi)set-collision resistance if for all ppt algorithms $\mathcal{A}(\cdot)$, any number c , and m large enough (with respect to c)¹,

$$\text{Prob} \left\{ \begin{array}{l} K \leftarrow \mathcal{K}_m, (S, M) \leftarrow \mathcal{A}(\mathcal{H}_K) : \\ S \text{ is a (multi)set and } M \text{ is a multiset of } B \\ \text{such that } S \neq M \text{ and } \mathcal{H}_K(S) \equiv_{\mathcal{H}_K} \mathcal{H}_K(M) \end{array} \right\} < m^{-c}.$$

Note that because $\mathcal{A}(\mathcal{H}_K)$ is polynomial in m_K , we will consider that it can only output polynomial sized S and M . We are disallowing compact representations for multisets that would allow $\mathcal{A}(\cdot)$ to express larger multisets (such compact representations do not lead to a feasible attack in our offline memory integrity application).

4 Additive Multiset Hash

In this section we give an example of a construction of (multi)set-collision resistant multiset hash functions. Let $B = \{0, 1\}^m$ represent the set of bit vectors of length m and let M be a multiset of elements of B . Recall that the number of times $b \in B$ is in the multiset M is denoted by M_b and is called the multiplicity of b in M . Let $H_K : \{0, 1\}^{m+1} \rightarrow \mathbb{Z}_n^l$ be randomly selected from a pseudorandom family of hash functions [9]. Let

$$L \approx n^l \approx 2^m, L \leq n^l, L \leq 2^m,$$

and define

$$\mathcal{H}_K(M) = \left[H_K(0, r) + \sum_{b \in B} M_b H_K(1, b) \mod n ; \sum_{b \in B} M_b \mod L ; r \right] \Bigg|_{r \leftarrow B},$$

where $r \in B$ is a random nonce². Notice that the logarithm of the cardinality m_K of the set into which \mathcal{H}_K maps multisets of B is equal to

$$m_K = \log(n^l) + \log(L) + \log(2^m) \approx 3m.$$

¹ The probability is taken over a random selection of K in \mathcal{K}_m (denoted by $K \leftarrow \mathcal{K}_m$) and over the randomness used in the ppt algorithm $\mathcal{A}(\mathcal{H}_K)$ (denoted by $(S, M) \leftarrow \mathcal{A}(\mathcal{H}_K)$).

² Note, the set from which r is taken could be smaller than B .

We say two triples $[h, c, r]$ and $[h', c', r']$ are equivalent, $[h; c; r] \equiv_{\mathcal{H}_K} [h'; c'; r']$, if and only if $h - H_K(0, r) = h' - H_K(0, r')$ modulo n and $c = c'$ modulo L . Notice that checking whether $\mathcal{H}_K(M) \equiv_{\mathcal{H}_K} \mathcal{H}_K(M')$ is efficient. We define addition of two triples $[h; c; r] +_{\mathcal{H}_K} [h'; c'; r']$ by the result of the computation

$$[H_K(0, r'') + h - H_K(0, r) + h' - H_K(0, r') \pmod n ; c + c' \pmod L ; r'']|_{r'' \leftarrow B}.$$

Clearly, $\mathcal{H}_K(M \cup M') \equiv_{\mathcal{H}_K} \mathcal{H}_K(M) +_{\mathcal{H}_K} \mathcal{H}_K(M')$, hence, $(\mathcal{H}_K, +_{\mathcal{H}_K}, \equiv_{\mathcal{H}_K})$ is a multiset hash. The proof of the next theorem is in Appendix A.

Theorem 1. *It is computationally infeasible to find a multiset M with multiplicities $< n$ and a multiset M' such that the cardinalities of M and M' are polynomial sized in m , $M \neq M'$, and $\mathcal{H}_K(M) \equiv_{\mathcal{H}_K} \mathcal{H}_K(M')$.*

As an example we consider $n = 2$ and $l = m$. Then the condition that a multiset M has multiplicities < 2 simply means that M is a set. This leads to set-collision resistance. Furthermore notice that addition modulo 2 defines xor \oplus .

Corollary 1. (*MSet-XOR-Hash*) *The multiset hash corresponding to*

$$\mathcal{H}_K(M) = \left[H_K(0, r) \oplus \bigoplus_{b \in B} M_b H_K(1, b) ; \sum_{b \in B} M_b \pmod{2^m} ; r \right]_{r \leftarrow B},$$

where $H_K : \{0, 1\} \times B \rightarrow \mathbb{Z}_2^m$ is randomly selected from a pseudorandom family of hash functions, is set-collision resistant.

Notice that $\mathcal{H}_K(M)$ keeps track of the cardinality of M . If this were not the case then $\mathcal{H}_K(S)$ and $\mathcal{H}_K(M)$ are equivalent for any S and M with $S_b = M_b$ modulo $n = 2$ for $b \in B$. This would contradict set-collision resistance. Also notice that $r \leftarrow B$ is randomly chosen. If r was a fixed known constant, then knowledge of n tuples $[M^i ; \mathcal{H}_K(M^i)]$ reveals n vectors

$$\bigoplus_{b \in B} M_b^i H_K(1, b) \in \mathbb{Z}_2^m.$$

If $n = 2m$ then with high probability these n vectors span the vector space \mathbb{Z}_2^m . This means that each vector in \mathbb{Z}_2^m can be constructed as a linear combination of these n vectors [4]:

$$\bigoplus_{i=1}^n a_i \cdot \left(\bigoplus_{b \in B} M_b^i H_K(1, b) \right) = \bigoplus_{b \in B} \left(\bigoplus_{i=1}^n a_i M_b^i \right) H_K(1, b).$$

Hence, a polynomial sized collision can be constructed for any polynomial sized M .

In Appendix B we show that for n exponentially large in m , we may remove the cardinality $\sum_{b \in B} M_b$ from the scheme altogether. By taking $l = 1$ and $L = n = 2^m$ we obtain the next corollary.

Corollary 2. (*MSet-Add-Hash*) *The multiset hash corresponding to*

$$\mathcal{H}_K(M) = \left[H_K(0, r) + \sum_{b \in B} M_b H_K(1, b) \mod 2^m ; r \right] \Bigg|_{r \leftarrow B},$$

where $H_K : \{0, 1\} \times B \rightarrow \mathbb{Z}_{2^m}$ is randomly selected from a pseudorandom family of hash functions, is multiset collision resistant.

The main difference between the **MSet-XOR-Hash** and **MSet-Add-Hash** is binary addition without and with carry respectively. This leads to either set collision resistance or multiset collision resistance.

In Appendix B we show that it is possible to replace the random nonce r by a counter that gets incremented on each use of \mathcal{H}_K . This removes the need for a random number generator from the scheme. Moreover, shorter values can be used for r as long as the key is changed when r overflows; this reduces the size of the hash. Also if the weighted sum of the hashes $H_K(1, b)$ in $\mathcal{H}_K(M)$ is never revealed to the adversary then we can remove $H_K(0, r)$ from the scheme altogether. For example, in the case where the weighted sums are encrypted by using a pseudorandom family of permutations (see Corollary 4 in Appendix B).

5 Multiplicative Multiset Hash

A multiset-collision resistant multiplicative multiset hash can be defined as follows. Let q be a large prime power and consider the computations in the field $GF(q)$. Let $H : B \rightarrow GF(q)$ be a poly-random function [9], that is, no polynomial time (in the logarithm of q) algorithm with oracle access H can distinguish between values of H and true random strings, even when the algorithm is permitted to select the arguments to H (in practice one would use MD5 [12] or SHA1 [11]). We define

$$\mathcal{H}(M) = \prod_{b \in B} H(b)^{M_b}, \quad (1)$$

$\equiv_{\mathcal{H}}$ to be equal to $=$, and $+\mathcal{H}$ to be multiplication in $GF(q)$.

Clearly, $(\mathcal{H}, +\mathcal{H}, \equiv_{\mathcal{H}})$ is a multiset hash. An advantage of the scheme is that we do not need a secret key. Unfortunately it relies on finite field arithmetic, which makes it too costly for some applications.

The proof of the following theorem is given in Appendix C, where we also define the discrete log (DL) assumption which says that for random $y \in GF(q)$ and generator $g \in GF(q)$, it is computationally infeasible to find x such that $g^x = y$ (x is called the discrete log of y).

Theorem 2. (*MSet-Mu-Hash*) *Under the DL assumption, the family³ of multiset hash functions, $(\mathcal{H}, +\mathcal{H}, \equiv_{\mathcal{H}})$, as defined in (1), is multiset collision resistant.*

³ The family is seeded by $GF(q)$.

Under certain assumptions we may replace multiplication in $GF(q)$ by addition modulo a large number. Even though the number of output bits of the resulting multiset hash needs to be much larger (since it is based on ‘weaker’ assumptions), the overall solution becomes more efficient since no finite field arithmetic is needed. Let $H : B \rightarrow \mathbb{Z}_n^l$, $n = 2^{\sqrt{m}}$, $l = \sqrt{m}$, be a poly-random function. Now, we define

$$\mathcal{H}(M) = \sum_{b \in B} M_b H(b) \mod n, \quad (2)$$

$\equiv_{\mathcal{H}}$ to be equal to $=$, and $+\mathcal{H}$ to be vector addition modulo n . See Appendix D for the proof of the next theorem and the definition of the worst-case shortest vector (SV) problem.

Theorem 3. (*MSet-VAdd-Hash*) *By assuming that the SV problem is infeasible to solve in polynomial time, the family⁴ of multiset hash functions, $(\mathcal{H}, +_{\mathcal{H}}, \equiv_{\mathcal{H}})$, as defined in (2), is multiset collision resistant.*

Remark. Because H can be evaluated with oracle access to \mathcal{H} , Theorems 2 and 3 still hold for a stronger form of multiset-collision resistance, in which it is computationally infeasible for an adversary with oracle access to H (instead of \mathcal{H}) to find a collision. This is what allows to use a publicly available H .

6 Integrity Checking of Random Access Memory

We now show how our multiset hash functions can be used to build secure offline integrity checkers for memory. Section 6.1 explains the model, and Section 6.2 shows our offline checker. Our implementation of this checker in the AEGIS secure processor [13] is described in [14,7].

6.1 Model

Figure 1 illustrates the model we use. There is a checker that keeps and maintains some small, fixed-sized, trusted state. The untrusted RAM (main memory) is arbitrarily large. The finite state machine (FSM) generates loads and stores and the checker updates its trusted state on each FSM load or store to the untrusted RAM. The checker uses its trusted state to verify the integrity of the untrusted RAM. The trusted computing base (TCB) consists of the FSM, and the checker with its trusted state. For example, the FSM could be a processor. The checker would be special hardware that is added to the processor to detect tampering in the external memory.

The checker checks if the untrusted RAM behaves correctly, i.e. like valid RAM. *RAM behaves like valid RAM if the data value that the checker reads from a particular address is the same data value that the checker had most recently*

⁴ The family is seeded by \mathbb{Z}_n^l .

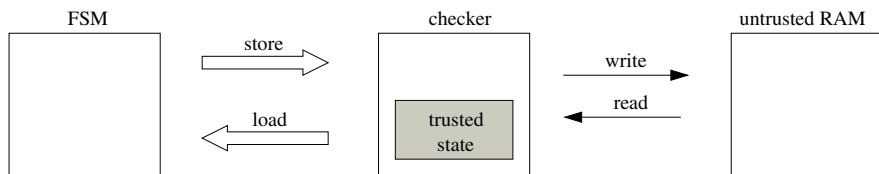


Fig. 1. Model

written to that address. In our model, the untrusted RAM is assumed to be actively controlled by an adversary. The untrusted RAM may not behave like valid RAM if the RAM has malfunctioned because of errors, or if it has been somehow altered by the adversary.

For this problem, a simple solution such as calculating a message authentication code (MAC) of the data value and address, writing the (data value, MAC) pair to the address, and using the MAC to check the data value on each read, does not work. The approach does not prevent replay attacks: an adversary can replace the (data value, MAC) pair currently at an address with a different pair that was previously written to the address. The essence of an offline checker is that a “log” of the sequence of FSM operations is maintained in fixed-sized trusted state in the checker.

6.2 Offline Checker

Figure 2 shows the basic **put** and **get** operations that are used internally in the checker. Figure 3 shows the interface the FSM calls to use the offline checker to check the integrity of the memory.

In Figure 2, the checker maintains two multiset hashes and a counter. In memory, each data value is accompanied by a time stamp. Each time the checker performs a **put** operation, it appends the current value of the counter (a time stamp) to the data value, and writes the (data value, time stamp) pair to memory. When the checker performs a **get** operation, it reads the pair stored at an address, and, if necessary, updates the counter so that it is strictly greater than the time stamp that was read. The multiset hashes are updated ($+_{\mathcal{H}}$) with (a, v, t) triples corresponding to the pairs written or read from memory.

Figure 3 shows how the checker implements the **store-load** interface. To initialize the RAM, the checker **puts** an initial value to each address. When the FSM performs a **store** operation, the checker **gets** the original value at the address, then **puts** the new value to the address. When the FSM performs a **load** operation, the checker **gets** the original value at the address and returns this value to the FSM; it then **puts** the same value back to the address. To check the integrity of the RAM at the end of a sequence of FSM stores and loads, the checker **gets** the value at each address, then compares **WRITEHASH** and **READHASH**. If **WRITEHASH** is equal to **READHASH**, the checker concludes that the RAM has been behaving correctly.

The checker's fixed-sized state is:

- 2 multiset hashes: WRITEHASH and READHASH. Initially both hashes are 0.
- 1 counter: TIMER. Initially TIMER is 0.

put(a, v) writes a value v to address a in memory:

1. Let t be the current value of TIMER. Write (v, t) to a in memory.
2. Update WRITEHASH: $\text{WRITEHASH} +_{\mathcal{H}} \text{hash}(a, v, t)$.

get(a) reads the value at address a in memory:

1. Read (v, t) from a in memory.
2. Update READHASH: $\text{READHASH} +_{\mathcal{H}} \text{hash}(a, v, t)$.
3. $\text{TIMER} = \max(\text{TIMER}, t + 1)$.

Fig. 2. put and get operations

Because the checker checks that WRITEHASH is equal to READHASH, substitution (the RAM returns a value that is never written to it) and replay (the RAM returns a stale value instead of the one that is most recently written) attacks on the RAM are prevented. The purpose of the time stamps is to prevent reordering attacks in which RAM returns a value that has not yet been written so that it can subsequently return stale data. Suppose we consider the **put** and **get** operations that occur on a particular address as occurring on a timeline. Line 3 in the **get** operation ensures that, for each **store** and **load** operation, each write has a time stamp that is strictly greater than all of the time stamps previously read from memory. Therefore, the first time an adversary tampers with a particular (data value, time stamp) pair that is read from memory, there will not be an entry in the WRITEHASH matching the adversary's entry in the READHASH, and that entry will not be added to the WRITEHASH at a later time.

The TIMER is not solely under the control of the checker, and is a function of what is read from memory, which is untrusted. Therefore, the WRITEHASH cannot be guaranteed to be over a set. For example, for a sequence of store and load operations occurring on the same address, an adversary can decrease the time stamp that is stored in memory and have triples be added to the WRITEHASH multiple times. The READHASH can also not be guaranteed to be over a set because the adversary controls the pairs that are read from memory. Thus, set-collision resistance is not sufficient, and we require multiset-collision resistant hash functions.

The proof of the following theorem is in Appendix E.

Theorem 4. *Let W be the multiset of triples written to memory and let R be the multiset of triples read from memory. That is, W hashes to WRITEHASH and R hashes to READHASH. Suppose the accesses to each address are an alternation of puts and gets. If the RAM does not behave like valid RAM, then $W \neq R$.*

```

initialize() initializes RAM.
    1. put( $a, 0$ ) for each address  $a$ .

store( $a, v$ ) stores  $v$  at address  $a$ .
    1. get( $a$ ).
    2. put( $a, v$ ).

load( $a$ ) loads the data value at address  $a$ .
    1.  $v = \mathbf{get}(a)$ . Return  $v$  to the caller.
    2. put( $a, v$ ).

check() checks if the RAM has behaved correctly (at the end of operation).
    1. get( $a$ ) for each address  $a$ .
    2. If WRITEHASH is equal to READHASH, return true.

```

Fig. 3. Offline integrity checking of random access memory

The following corollary shows the hardness of breaking our offline memory integrity checking scheme.

Corollary 3. *Tampering with the RAM without being detected is as hard as finding a collision $W \neq R$ for the multiset hash function.*

Offline memory integrity checking was introduced by Blum et al. [6]. However, the original offline checker in [6] differs from our checker in two respects. First, the original checker is implemented with ϵ -biased hash functions [10]. These hash functions are set-collision resistant against random errors but not against a malicious adversary. Secondly, the **TIMER** is incremented on each **put** operation and is not a function of what is read from memory. The **TIMER** is solely under the control of the checker. This means that the pairs that are used to update **WRITEHASH** form a set. Therefore set-collision resistance is sufficient. The original offline checker can be made secure against active adversaries by using a set-collision resistant multiset hash function, instead of ϵ -biased hash functions. Our offline checker improves on the original checker because **TIMER** is not incremented on every **load** and **store** operation. Thus, time stamps can be smaller without increasing the frequency of checks, which improves the performance of the checker.

7 Conclusion

We have introduced incremental multiset hash functions which can be efficiently updated, and for which the ordering of inputs is not important. Table 1 summarizes our comparison of the multiset hash functions introduced in this paper.

Table 1. Comparison of the Multiset Hash Functions

	collision resistance	key	security based on	comput. efficiency	length of output	offline checker	hash visible
MSet-XOR-Hash	set	Y	PRF	++	+	original	r/enc
MSet-Add-Hash	multiset	Y	PRF	++	+	both	r/enc
MSet-Mu-Hash	multiset	N	RO/DL	—	+	both	
MSet-VAdd-Hash	multiset	N	RO/SV	+	—	both	

In the table, we indicate whether the security is based on pseudorandom family of hash functions (PRF), the random oracle model (RO), the discrete log assumption (DL), or/and the hardness of the worst case shortest vector problem (SV). If hashes are to be visible to the adversary (i.e., the adversary can see the hashes in the trusted state, but cannot modify them), we indicate whether a random nonce/counter (r), or encryption is necessary. We have improved the security and the performance of the offline memory integrity checker in [6] as one application of these functions.

References

1. M. Ajtai. Generating hard instances of lattice problems. In *28th ACM STOC*, pages 99–108, 1996.
2. M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography: The case of hashing and signing. In *Crypto '94*, LNCS 839. Springer-Verlag, 1994.
3. M. Bellare, R. Guerin, and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In *Crypto '95*, LNCS 963. Springer-Verlag, 1995.
4. M. Bellare and D. Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In *Eurocrypt '97*, LNCS 1233. Springer-Verlag, 1997.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS '93*, pages 62–73. ACM Press, 1993.
6. M. Blum, W. S. Evans, P. Gemmell, S. Kannan, and M. Naor. Checking the correctness of memories. In *Algorithmica*, volume 12, pages 225–244, 1994.
7. D. Clarke, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. Offline integrity checking of untrusted storage. In *MIT-LCS-TR-871*, Nov. 2002.
8. O. Goldreich, S. Goldwasser, and S. Halevi. Collision-free hashing from lattice problems. In *Theory of Cryptography Library 96-09*, July 1996.
9. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):210–217, 1986.
10. J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *22nd ACM STOC*, pages 213–223, 1990.
11. NIST. FIPS PUB 180-1: Secure Hash Standard, April 1995.
12. R. Rivest. RFC 1321: The MD5 Message-Digest Algorithm, April 1992.
13. G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas. AEGIS: Architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th Int'l Conference on Supercomputing*, June 2003.
14. G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas. Efficient memory integrity verification and encryption for secure processors. In *Proceedings of the 36th Int'l Symposium on Microarchitecture*, Dec 2003.

A Proof of Collision Resistance of Additive Hash

Let \mathcal{G}_m be the family of matrices with 2^{m+1} rows, l columns, and entries in \mathbb{Z}_n (recall $L \approx n^l \approx 2^m$). Let H_K be a random matrix in $\mathcal{G}_m = \{H_1, H_2, H_3, \dots\}$. Notice that H_K is the K -th matrix in \mathcal{G}_m . We assume that this matrix, or equivalently its label K , is secret and only accessible by the secure processor. The family of matrices \mathcal{G}_m from which H_K is selected is publicly known.

The rows of H_K are labelled by $x \in \{0, 1\}^{m+1}$ and denoted by $H_K(x)$. This represents H_K as a function from $x \in \{0, 1\}^{m+1}$ to \mathbb{Z}_n^l , the set of vectors with length l and entries in \mathbb{Z}_n . In practice, H_K is not a completely random matrix over \mathbb{Z}_n , but H_K is selected from a pseudorandom family of functions. We address this issue as soon as we are ready to formulate a proof of Theorem 1.

The following theorem is about the probability that an adversary finds a collision for some multiset M' . The probability is taken over random matrices H_K in \mathcal{G}_m ($H_K \leftarrow \mathcal{G}_m$) and the randomness of the random nonce used in \mathcal{H}_K .

Theorem 5. *Let M and M' be multisets of B . Let d be the greatest common divisor⁵ of n and each of the differences $|M_b - M'_b|$, $b \in B$. Given knowledge of u tuples $[M^i ; \mathcal{H}_K(M^i)]$, the probability that M is a collision for M' is at most $u^2/2^m + (d/n)^l$.*

We first introduce some notation. Let $v(r, M)$ be the vector of length 2^{m+1} defined by

$$v(r, M)_{(0,b)} = 1 \text{ if and only if } b = r$$

and

$$v(r, M)_{(1,b)} = M_b.$$

Let $v(M)$ be the vector of length 2^{m+1} defined by $v(M)_{(0,b)} = 0$ and $v(M)_{(1,b)} = M_b$.

Lemma 1. *(i) Knowing $[M ; \mathcal{H}_K(M)]$ is equivalent to knowing*

$$[v(r, M) ; v(r, M)H_K \pmod n].$$

(ii) $\mathcal{H}_K(M) \equiv_{\mathcal{H}_K} \mathcal{H}_K(M')$ if and only if $v(M)H_K = v(M')H_K$ modulo n and $\sum_{b \in B} M_b = \sum_{b \in B} M'_b$ modulo L .

Proof. Notice that $v(r, M)$ encodes r , M , and, hence, the cardinality $\sum_{b \in B} M_b$ of M , and notice that

$$\mathcal{H}_K(M) = \left[v(r, M)H_K \pmod n ; \sum_{b \in B} M_b \pmod L ; r \right].$$

The lemma follows immediately from these observations.

⁵ The greatest common divisor of 0 with a positive integer i is equal to i .

Suppose that an adversary learns u tuples $[M^i ; \mathcal{H}_K(M^i)]$ or, according to Lemma 1.(i), u vectors $v(r^i, M^i)$ together with the corresponding $v(r^i, M^i)H_K$ modulo n . Let A be the $u \times 2^{m+1}$ matrix with rows $v(r^i, M^i)$. Then the matrix with rows $v(r^i, M^i)H_K$ is equal to AH_K . Clearly, A modulo n has full rank over \mathbb{Z}_n if all r^i are different. The probability that there are two equal r^i 's is at most $u^2/2^m$.

Lemma 2. *The probability that the r^i 's corresponding to matrix A are all different is at least $1 - u^2/2^m$.*

By Lemma 1.(ii), in order to find a collision for M' , the adversary needs to find a multiset $M \neq M'$ such that $v(M)H_K = v(M')H_K$ modulo n and such that the cardinalities of M and M' are equal to one another modulo L . The next three lemmas show how difficult this is for the adversary if he is in the situation of the previous lemma.

Lemma 3. *Let M and M' be multisets of B . The probability that $v(M)H_K = v(M')H_K$ modulo n is statistically independent of the knowledge of a full rank matrix A over \mathbb{Z}_n corresponding to different r^i 's and the knowledge of $h = AH_K$ modulo n .*

Proof. W.l.o.g. (after reordering the columns of A and the corresponding entries of $v(M) - v(M')$ and corresponding rows of H_K) matrix A has the form $A = (I \ A^1)$, where I is the $u \times u$ identity matrix, and $v(M) - v(M')$ has the form $(0 \ v)$, where 0 is the all zero vector of length u . Denote the top u rows of H_K by H_K^0 and let H_K^1 be such that

$$H_K = \begin{pmatrix} H_K^0 \\ H_K^1 \end{pmatrix}.$$

Clearly, the equation $h = AH_K$ modulo n is equivalent to

$$h = H_K^0 + A^1 H_K^1 \pmod{n}. \quad (3)$$

The equation $0 = (v(M) - v(M'))H_K$ modulo n is equivalent to

$$0 = v H_K^1 \pmod{n}. \quad (4)$$

Straightforward counting tells us that $\text{Prob}\{(4)|(3)\}$ is equal to the # of matrices H_K^1 satisfying (4) divided by the total # of matrices H_K^1 . This is in turn equal to the # of matrices H_K satisfying (4) divided by the total # of matrices H_K , which is $\text{Prob}\{(4)\}$.

Lemma 4. *Let M and M' be multisets of B . Let d be the greatest common divisor of n and each of the differences $|M_b - M'_b|$, $b \in B$. Then $(v(M) - v(M'))H_K$ modulo n is uniformly distributed in $d\mathbb{Z}_n^l$.*

Proof. To prove this lemma, we show that each entry of $(v(M) - v(M'))H_K$ modulo n is uniformly distributed in $d\mathbb{Z}_n$. Let y represent one of the columns of H_K and define for $\beta \in \mathbb{Z}_n$ the set

$$\mathcal{C}_\beta = \{y : (v(M) - v(M'))y = \beta \pmod n\}.$$

Since d divides each entry of $v(M) - v(M')$, it also divides the product $(v(M) - v(M'))y$, hence, $\mathcal{C}_\beta = \emptyset$ if β is not divisible by d . Since d is the greatest common divisor of n and each of the entries of $v(M) - v(M')$, there exists a vector y such that $(v(M) - v(M'))y = d$ modulo n . This proves that $\mathcal{C}_\beta \neq \emptyset$ if and only if d divides β . For a fixed column $y' \in \mathcal{C}_\beta \neq \emptyset$, the mapping $y \in \mathcal{C}_\beta \rightarrow y - y' \pmod n$ is a bijection. Hence, the non-empty sets \mathcal{C}_β have equal cardinality. We conclude that each entry of $(v(M) - v(M'))H_K$ modulo n is uniformly distributed in $d\mathbb{Z}_n$.

Lemma 5. *Let M and M' be multisets of B . Let d be the greatest common divisor of n and each of the differences $|M_b - M'_b|$, $b \in B$. Given knowledge of a full rank matrix A over \mathbb{Z}_n corresponding to different r^i 's and given knowledge of $h = AH_K$ modulo n , the probability that $v(M)H_K = v(M')H_K$ modulo n is equal to $(d/n)^l$.*

Proof. By Lemma 3, since matrix A corresponds to different r^i 's and $(v(M) - v(M'))_{(0, r^i)} = 0$, the probability that the randomly chosen matrix H_K satisfies $0 = (v(M) - v(M'))H_K$ modulo n is independent of the knowledge of $h = AH_K \pmod n$. By Lemma 4, since H_K is uniformly distributed, $(v(M) - v(M'))H_K$ is uniformly distributed in $d\mathbb{Z}_n^l$. Hence, the probability that $0 = (v(M) - v(M'))H_K \pmod n$ is equal to one divided by the cardinality of $d\mathbb{Z}_n^l$, which is equal to $(d/n)^l$.

Combining Lemmas 2 and 5 proves Theorem 5. To prove Theorem 1 we need the following extra lemma.

Lemma 6. *Suppose that $v(M) = v(M')$ modulo n , $\sum_{b \in B} M_b = \sum_{b \in B} M'_b$ modulo L , the cardinalities of M and M' are $< L$, and that the multiplicities of M are $< n$. Then $M = M'$.*

Proof. If the cardinalities of M and M' are equal modulo L and $< L$ then

$$\sum_{b \in B} M_b = \sum_{b \in B} M'_b. \quad (5)$$

If all entries of $v(M)$ are $< n$ and $v(M) = v(M')$ modulo n , then

$$M'_b = M_b + \beta_b n, \quad b \in B, \quad (6)$$

for integers $\beta_b \geq 0$. Combining (5) and (6) proves $\sum_{b \in B} \beta_b = 0$, hence, all $\beta_b = 0$. We conclude that $M = M'$.

Now we are ready to prove Theorem 1.

Proof. Let $\mathcal{A}(\mathcal{H}_K)$ be a probabilistic polynomial time (in $m_K \approx 3m$) algorithm with oracle access to $(\mathcal{H}_K, +_{\mathcal{H}_K}, \equiv_{\mathcal{H}_K})$. Then $\mathcal{A}(\mathcal{H}_K)$ can gain knowledge about at most a polynomial number $u(m)$ tuples $[M^i ; \mathcal{H}_K(M^i)]$ (here $u(\cdot)$ denotes a polynomial). Furthermore, $\mathcal{A}(\mathcal{H}_K)$ can search for a collision among at most a polynomial number $t(m)$ of pairs (M, M') , where M and M' are multisets, $M \neq M'$, and M has multiplicities $< n$. According to Theorem 5, the probability that $\mathcal{A}(\mathcal{H}_K)$ finds a collision is at most

$$t(m)(u(m)^2/2^m + (d/n)^l).$$

Since $\mathcal{A}(\mathcal{H}_K)$ can only compute polynomial sized multisets, the cardinality of the multisets M and M' are $< L \approx 2^m$. This allows us to apply Lemma 6 and conclude that $0 \neq (v(M) - v(M')) \bmod n$. Hence, the greatest common divisor d of n and each of the differences $|M_b - M'_b|$, $b \in B$, is at most $n/2$. This leads to

$$(d/n)^l \leq 2^{-l}.$$

Let $c > 0$ be any number and suppose that $2^{-l} \geq m^{-c}$, or equivalently, $l \leq c \log m$. Notice that each of the differences $|M_b - M'_b|$ is polynomial sized in m , hence, d is polynomial sized in m and there exists a number $e > 0$ such that $d \leq m^e$ for m large enough. This proves

$$(d/n)^l \leq m^{el}/n^l \approx m^{el}/2^m \leq m^{ec \log m}/2^m,$$

which is at most m^{-c} for m large enough. We conclude that the probability that $\mathcal{A}(\mathcal{H}_K)$ finds a collision is at most m^{-c} for m large enough. This proves Theorem 1 for random matrices H_K .

Remark. The theorem also holds for a pseudorandom family of hash functions represented as matrices. Suppose that an adversary can compute a collision with a significant probability of success in the case where a pseudorandom family of hash functions is used. We have just shown that an adversary has a negligible probability of success in the case where random hash functions are used. Hence, with a significant probability of success he is able to distinguish between the use of pseudorandom hash functions and the use of random hash functions. This contradicts the definition of pseudorandomness, see [3] for a detailed proof of a similar result.

B Variants of Additive Hash

A few interesting variants of \mathcal{H}_K exist. Suppose that $v(M) = v(M')$ modulo n and that the multiplicities of M and M' are $< n$. Then clearly $M = M'$. Hence, we do not need Lemma 6 in the proof of Theorem 5. This means that the proof of Theorem 5 does not depend on the cardinalities of M and M' to be equal modulo L . We can remove the cardinality $\sum_{b \in B} M_b$ from the scheme altogether. For example, for n exponentially large, the cardinalities and in particular the

multiplicities of M and M' are $< n$. This proves Corollary 2. An other example is $n = 2$ and both M and M' are sets, which proves the main result of [3].

Secondly, it is possible to replace the random nonce r by a counter that gets incremented on each use of \mathcal{H}_K , or by any other value that never repeats itself in polynomial time. This guarantees with probability 1 that the matrix A corresponds to different r^i 's (see Lemma 2). This removes the need for a random number generator from the scheme. Moreover, shorter values can be used for r as long as the key is changed when r overflows; this reduces the size of the hash.

If $u = 0$ then the proof of Theorem 5 does not depend on matrix A and its corresponding r^i 's. Similarly, if sums of hashes,

$$H_K(0, r) + \sum_{b \in B} M_b H_K(1, b) \pmod n,$$

are hidden from the adversary (he knows which multiset M is being hashed, but not the value of the sum of hashes) then we can remove $H_K(0, r)$ from the scheme altogether. As the following corollary shows, complete hiding is not necessary. We can use a pseudorandom permutation to hide sums of hashes.

Corollary 4. (*Permuted-MSet-XOR-Hash*) The multiset hash corresponding to

$$\mathcal{H}_{K,K'}(M) = \left[P_{K'} \left(\bigoplus_{b \in B} M_b H_K(1, b) \right) ; \sum_{b \in B} M_b \pmod{2^m} \right],$$

where $H_K : \{0, 1\} \times B \rightarrow \mathbb{Z}_2^m$ and $P_{K'}$ are randomly selected from a pseudorandom family of hash functions and permutations, is set-collision resistant.

(*Permuted-MSet-Add-Hash*) The multiset hash corresponding to

$$\mathcal{H}_{K,K'}(M) = P_{K'} \left(\sum_{b \in B} M_b H_K(1, b) \pmod{2^m} \right)$$

where $H_K : \{0, 1\} \times B \rightarrow \mathbb{Z}_{2^m}$ and $P_{K'}$ are randomly selected from a pseudorandom family of hash functions and permutations, is multiset-collision resistant.

Notice that the multiset hashes are incremental because $P_{K'}$ is a permutation and, hence, invertible.

Proof. We first consider a random function $P_{K'}$. Suppose that the adversary learns u tuples $[M^i ; \mathcal{H}_{K,K'}(M^i)]$. As in Lemma 2, the probability that two permuted sums of hashes in the u tuples are equal is at most $u^2/2^m$. If all of them are unequal to one another then matrix AH_K (defined without the part corresponding to the random nonce) is uniformly distributed and not known to the adversary (since $P_{K'}$ is a random function). Hence, the probability that $v(M)H_K = v(M')H_K$ modulo n is statistically independent of the knowledge of the adversary. This can be used instead of Lemma 5 to prove Theorems 5 and 1. This result also holds for a pseudorandom family of permutations $P_{K'}$, see the remark at the end of the proof of Theorem 1 in Appendix A.

C Proof of Collision Resistance of Multiplicative Hash

In the following lemma $\mathcal{A}(\cdot)$ is a probabilistic polynomial time (in $\log q$) algorithm which outputs weights⁶ $w_1, \dots, w_u \in \mathbb{Z}_{q-1}$ for a polynomial number of random inputs $x_1, \dots, x_u \in GF(q)$ such that $1 = \prod_i x_i^{w_i}$ with probability at least ρ . We show that if such an algorithm exists then we can break the DL problem in $GF(q)$ in polynomial time with probability at least ρ .

Lemma 7. *Let $\mathcal{A}(\cdot)$ be a ppt algorithm such that there exists a number c such that for $u \leq (\log q)^c$,*

$$\text{Prob} \left\{ (x_i \leftarrow GF(q))_{i=1}^u, (w_i \in \mathbb{Z}_{q-1})_{i=1}^u \leftarrow \mathcal{A}(x_1, \dots, x_u) : \right. \\ \left. 1 = \prod_i x_i^{w_i}, \exists_i w_i \neq 0, \forall_i |w_i| \leq (\log q)^c \right\} \geq \rho. \quad (7)$$

Let g be a generator of $GF(q)$. Then there exists a probabilistic polynomial time (in $\log q$) algorithm $\mathcal{A}'(\cdot)$ such that

$$\text{Prob}\{y \leftarrow GF(q), x \leftarrow \mathcal{A}'(y) : y = g^x\} \geq \rho/(\log q)^c.$$

In words, given a random $y \in GF(q)$, we are able to find the discrete log of y in $GF(q)$ with probability at least $\rho/(\log q)^c$.

Proof. Let $y \leftarrow GF(q)$. Select a polynomial number u of random elements r_1, \dots, r_u in \mathbb{Z}_{q-1} and $j \in \{1, \dots, u\}$ and compute

$$x_j = yg^{r_j} \text{ and } x_i = g^{r_i} \text{ for } i \neq j.$$

Compute $(w_1, \dots, w_u) \leftarrow \mathcal{A}(x_1, \dots, x_u)$. Since by construction the x_i s have been chosen uniformly at random, we know that with probability at least ρ the weights $w_1, \dots, w_u \in \mathbb{Z}_{q-1}$ are computed such that they are not all equal to zero, $|w_j| \leq (\log q)^c$, and

$$1 = \prod_i x_i^{w_i} = y^{w_j} g^{\sum_i r_i w_i}. \quad (8)$$

Since the u inputs are in random order, the probability that $w_j \neq 0$ is at least

$$1/u \geq (\log q)^{-c}.$$

Suppose that $w_j \neq 0$. Let d be the greatest common divisor between w_j and $q-1$. Then⁷ w_j/d is invertible in \mathbb{Z}_{q-1} . By using the Chinese remainder theorem (assuming that we know the factorization of $q-1$), we are able to compute the inverse of w_j/d in \mathbb{Z}_{q-1} in polynomial time. Denote this inverse by w'_j . From (8) we infer that

$$y^d = g^{-w'_j \sum_i r_i w_i}.$$

⁶ Not all equal to zero and each of them bounded by a polynomial number.

⁷ Division $/$ denotes division over integers, not over \mathbb{Z}_{q-1} (since d has no inverse in \mathbb{Z}_{q-1} , we can not divide w_j by d in \mathbb{Z}_{q-1}).

Notice that if $y^d = g^s$ and $y = g^t$, then $g^{dt} = g^s$, that is $dt = s$ modulo $q-1$. Recall that d divides $q-1$. For this reason d must also divide s . Let $d' = (q-1)/d$ and $s' = s/d$. Both can be computed in polynomial time as we have shown. Now y can be expressed as one of the roots

$$y = g^{s'+jd'},$$

where $0 \leq j \leq d-1$. Since $d \leq |w_j| \leq (\log q)^c$, each of the roots can be checked in polynomial time. This proves the lemma.

The DL assumption states that for all ppt algorithms $\mathcal{A}(\cdot)$, any number c , and Q large enough,

$$\text{Prob} \left\{ q \geq Q \text{ is a prime power, } g \text{ generates } GF(q), : y = g^x \right\} \leq (\log q)^{-c}.$$

We are ready to prove Theorem 2.

Proof. Suppose that there exists a number c and a probabilistic polynomial time algorithm $\mathcal{B}(H)$, which runs in time $u = (\log q)^c$, with access to a random oracle H which outputs with probability $\rho \geq 1/u$ a collision M for M' . That is, $M \neq M'$, M and M' are polynomial sized $< u$, and

$$\mathcal{H}(M) = \prod_{b \in B} H(b)^{M_b} = \prod_{b \in B} H(b)^{M'_b} = \mathcal{H}(M').$$

This means that

$$1 = \prod_{b \in B} H(b)^{M_b - M'_b},$$

there is a polynomial number M_b 's and M'_b 's unequal to zero, for all $b \in B$ the absolute value $|M_b - M'_b| < u$ is polynomial sized, and there exists a $b \in B$ such that $M_b - M'_b \neq 0$.

Let \mathcal{C} be an algorithm that goes from $GF(q)^u$ to $B \rightarrow GF(q)$, where $B \rightarrow GF(q)$ denotes the set of oracles with inputs in B and outputs in $GF(q)$. \mathcal{C} is chosen such that $\mathcal{C}(x_1, \dots, x_u)$ returns x_1 when it is called for the first time on some input y_1 , x_2 when it is called for the first time on some input y_2 different from y_1 , and so on.

When x_1, \dots, x_u are chosen randomly, $\mathcal{C}(x_1, \dots, x_u)$ cannot be distinguished from a random oracle by \mathcal{B} because \mathcal{B} cannot query \mathcal{C} more than u times. Therefore, if we let \mathcal{A} be the composition of \mathcal{B} and \mathcal{C} , \mathcal{A} is able to find a collision for \mathcal{H} with probability ρ when its inputs are chosen uniformly at random. Moreover, \mathcal{A} is a ppt algorithm satisfying (7), so by Lemma 7, \mathcal{A} can break the discrete log problem in $GF(q)$ in polynomial time with probability at least $\rho/(\log q)^c \geq (\log q)^{-2c}$. This contradicts the DL assumption. So \mathcal{B} does not exist, which proves multiset-collision resistance.

Because oracle access to H is stronger than oracle access to \mathcal{H} , this proves Theorem 2 when H is a random oracle. The result carries over to poly-random functions because they are indistinguishable from random functions by ppt algorithms.

Remark. Supposing that H is a random oracle is a strong assumption. Compared to the **MSet-XOR-Hash** and **MSet-Add-Hash** we do not need a secret key (as the seed of a pseudorandom family of hash functions) at all. We refer to [5] for a discussion into what extent the random oracle assumption can be met in practice.

D Proof of Collision Resistance of Vector Additive Hash

If r is a fixed constant in the **MSet-Add-Hash**, then we are again vulnerable for the attack described for the **MSet-XOR-Hash**, where r is a fixed constant. The main difference is that the attack is not modulo $n = 2$ but modulo $n = 2^m$. This means that the linear combination may lead to a collision with large multiplicities. This would give a non-polynomial sized collision and does not defeat the multiset collision resistance. It turns out that this problem is related to a weighted knapsack problem (see also [4]). In this sense **MSet-Add-Hash** remains multiset collision resistant, even if the pseudorandom family of hash functions H_K is replaced by a single random function avoiding the use of a secret key as in **MSet-Mu-Hash**.

The weighted knapsack (WK) assumption is defined as follows. For all ppt algorithms $\mathcal{A}(\cdot)$, any number c , q large enough, and $u \leq (\log q)^c$,

$$\text{Prob} \left\{ \begin{array}{l} (x_i \leftarrow \mathbb{Z}_q)_{i=1}^u, (w_i \in \mathbb{Z}_q)_{i=1}^u \leftarrow \mathcal{A}(x_1, \dots, x_u) : \\ 0 = \sum_i w_i x_i \pmod{q}, \exists_i w_i \neq 0, \forall_i |w_i| \leq (\log q)^c \end{array} \right\} \leq (\log q)^{-c}.$$

Notice the resemblance with (7), where multiplication in $GF(q)$ is now replaced by addition modulo q (where q can be any integer and does not need to be a prime power). It remains unclear to what extent Ajtai's work [1] relates this problem to the *worst-case* shortest vector problem. It is an open problem whether to believe in the WK assumption.

Let $H : B \rightarrow \mathbb{Z}_q$ be a poly-random function. We define

$$\mathcal{H}(M) = \sum_{b \in B} M_b H(b) \pmod{q}, \quad (9)$$

$\equiv_{\mathcal{H}}$ to be equal to $=$, and $+\mathcal{H}$ to be addition modulo q (q plays the role of 2^m in **MSet-Add-Hash**). The proof of the next theorem is similar to the proof of Theorem 2 in Appendix C.

Theorem 6. *Under the WK assumption, $(\mathcal{H}, +_{\mathcal{H}}, \equiv_{\mathcal{H}})$ as defined in (9) is multiset collision resistant.*

For completeness, we introduce a multiset hash corresponding to parameters $n = 2^{\sqrt{m}}$ and $l = \sqrt{m}$ (see Section 4). Let $H : B \rightarrow \mathbb{Z}_n^l$ be a poly-random function. Now, we define

$$\mathcal{H}(M) = \sum_{b \in B} M_b H(b) \pmod{n},$$

$\equiv_{\mathcal{H}}$ to be equal to $=$, and $+_{\mathcal{H}}$ to be vector addition modulo n . Theorem 6 holds again if we modify the WK assumption by replacing $x_i \leftarrow \mathbb{Z}_q$ by $x_i \leftarrow \mathbb{Z}_n^l$, $w_i \in \mathbb{Z}_q$ by $w_i \in \mathbb{Z}_n$, and q by n . The main difference is that the x_i 's are vectors of length $l = \sqrt{m}$. According to [8, Sections 2.1 and 2.2]⁸, if there is a ppt solving the modified WK problem (that is it contradicts the modified WK assumption) then, by Ajtai's theorem [1], there is a probabilistic polynomial (in l) algorithm which, for *any* lattice \mathcal{L} in \mathbb{R}^l , given an arbitrary basis of \mathcal{L} , approximates (up to a polynomial factor in l) the length of the shortest vector in \mathcal{L} . This proves Theorem 3. The worst-case shortest vector problem is believed to be hard, see [8] for more discussion.

E Proof of Improved Offline Checker

In this appendix, we prove Theorem 4.

Proof. Suppose the RAM does not behave like valid RAM (i.e. the data value that the checker reads from an address is not the same data value that the checker had most recently written to that address). We will prove that $W \neq R$.

Consider the **put** and **get** operations that occur on an address as occurring on a timeline. To avoid confusion with the values of **TIMER**, we express this timeline in terms of processor cycles. Let x_1 be the cycle of the first incorrect **get** operation. Suppose the checker reads the pair (v_1, t_1) from address a at x_1 . If there does not exist a cycle at which the checker writes the pair (v_1, t_1) to address a , then $W \neq R$ and we are done.

Suppose there is a cycle x_2 when the checker first writes (v_1, t_1) to address a . Because of line 3 in the **get** operation, the values of time stamps of all of the writes to a after x_1 are strictly greater than t_1 . Because the time stamps at x_1 and x_2 are the same, and since **put** operations and **get** operations do not occur on the same cycle, x_2 occurs before x_1 ($x_2 < x_1$). Let x_3 be the cycle of the first read from a after x_2 . Notice that x_1 is a read after x_2 , so $x_1 \geq x_3$. If x_1 were equal to x_3 , then the data value most recently written to a , i.e. v_1 , would be read at x_1 . This contradicts the assumption that x_1 is an incorrect read. Therefore, $x_1 > x_3$.

Because the read at cycle x_1 is the first incorrect read, the read at cycle x_3 is a correct read. So the read at x_3 reads the same pair that was written at x_2 . Again, because of line 3 in the **get** operation, the values of time stamps of all the writes to a after x_3 are strictly greater than t_1 . Therefore, (v_1, t_1) cannot be written after x_3 . Because x_2 is the first cycle on which (v_1, t_1) is written to a , (v_1, t_1) cannot be written before x_2 . Because x_3 is the first read from a after x_2 , and two writes to an address always have a read from that address between them, (v_1, t_1) cannot be written between x_2 and x_3 . Therefore, the pair (v_1, t_1) is written only once, but it is read at x_1 and x_3 . Therefore, $W \neq R$.

⁸ Notice that the matrix with columns x_i is in $\mathbb{Z}_n^{l \times u}$ and that the vector with entries w_i is unequal to zero and has Euclidean norm polynomial in $l = \sqrt{m}$.

New Parallel Domain Extenders for UOWHF

Wonil Lee¹, Donghoon Chang¹, Sangjin Lee¹,
Soohak Sung², and Mridul Nandi³

¹ Center for Information and Security Technologies
Korea University, Seoul, Korea
`{wonil,dhchang,sangjin}@cist.korea.ac.kr`

² Applied Math. Department, Paichai University, Daejeon, Korea
`sungsh@mail.paichai.ac.kr`

³ Applied Statistics Unit, Indian Statistical Institute, Kolkata, India
`hi_mridul@yahoo.com`

Abstract. We present two new parallel algorithms for extending the domain of a UOWHF. The first algorithm is complete binary tree based construction and has less key length expansion than Sarkar's construction which is the previously best known complete binary tree based construction. But only disadvantage is that here we need more key length expansion than that of Shoup's sequential algorithm. But it is not too large as in all practical situations we need just two more masks than Shoup's. Our second algorithm is based on non-complete l -ary tree and has the same optimal key length expansion as Shoup's which has the most efficient key length expansion known so far. Using the recent result [9], we can also prove that the key length expansion of this algorithm and Shoup's sequential algorithm are the minimum possible for any algorithms in a large class of "natural" domain extending algorithms. But its parallelizability performance is less efficient than complete tree based constructions. However if l is getting larger, then the parallelizability of the construction is also getting near to that of complete tree based constructions. We also give a sufficient condition for valid domain extension in sequential domain extension.

Keywords: UOWHF, hash function, masking assignment, sequential construction, parallel construction, tree based construction.

1 Introduction

Naor and Yung [7] introduced the notion of universal one-way hash function (UOWHF) to prove that secure digital signatures can be based on any 1-1 one-way function. A UOWHF is a family of functions $\{h_k\}_{k \in \mathcal{K}}$ for which the following task of the adversary is computationally infeasible. The adversary has to choose a x from the domain, and then given a random $k \in \mathcal{K}$, he has to find a y such that $x \neq y$ but $h_k(x) = h_k(y)$. Intuitively, a UOWHF is a weaker primitive than a collision resistant hash function (CRHF), since the task of the adversary is more difficult, i.e., the adversary has to commit to the string x before knowing the actual hash function h_k for which the collision has to be found. Furthermore,

Simon [11] had shown that there is an oracle relative to which UOWHFs exist but not CRHFs.

A UOWHF is an attractive alternative to a CRHF because it seems that building an efficient and secure UOWHF is easier than building an efficient and secure CRHF, and in many applications, most importantly for building digital signature schemes, a UOWHF is sufficient. In addition, as mentioned in [1], the birthday attack does not apply to UOWHFs. Hence the size of the message digest can be significantly shorter.

A reasonable approach to designing a UOWHF that hashes messages of arbitrary and variable length is to first design a compression function, that is, a UOWHF that hashes fixed-length messages, and then design a method for composing these compression functions so as to hash arbitrary and variable messages. The present paper deals with the second problem, that of composing compression functions. We will call the composite method *construction* or *domain extender* for the most part in this paper. The main technical problem in designing such domain extender is to keep the key length of the domain extender from getting too large.

The rest of this paper is organized as follows. Motivation and our contributions are given in Section 2. Some detailed history of UOWHF is also provided in Section 2 in order to precisely explain our contributions. Preliminaries are given in Section 3. We will generalize Shoup’s sequential construction in Section 4. In this section we also provide a sufficient condition for valid sequential domain extension. Then we will present our new complete binary tree based parallel domain extender and will give a proof of validness of the extension in Section 5. we will present our second new parallel domain extender which is based on non-complete l -ary tree and the proof of security in Section 6. In Section 7, we specifically compare the known constructions with our two constructions. This paper concludes with Section 8.

2 Motivation and Our Contribution

Most practical signature schemes follow “hash-and-sign” paradigm. They take a message M of an arbitrary length and hash it to obtain a constant length string, which is then fed into a signing algorithm. Many schemes use CRHFs to hash a message x , but as it was first pointed out in [1] a UOWHF suffices for that purpose. Indeed, if $\{h_k\}_{k \in \mathcal{K}}$ is a UOWHF, then to sign a message x , the signer chooses a random key k , and produces the signature $(k, \sigma(k, h_k(x)))$, where σ is the underlying signing function for short messages.

Note that the key length varies with the length of input message for UOWHFs. Therefore, in many cases, the size of $(k, h_k(x))$ can be larger than the input size of σ . However, in these cases, we can solve the problem by applying the signing algorithm σ to $(h_{K'}(k), h_k(x))$, where K' is part of the signer’s public key. Here the signature becomes $(k, \sigma(h_{K'}(k), h_k(x)))$. And note that the function $h_{K'}$ can be replaced by any second-preimage resistant function, because its input is random and chosen by the signer. Since messages can be very long,

hashing speed is a crucial factor. On the other hand, a closer look at the signature scheme reveals that the key k must be part of the signature so the receiver can recompute the hash. Therefore the shorter the key better the signature scheme.

These facts lead us to think we should consider two aspects.

1. Minimizing the key length expansion: This is certainly a very important aspect of any domain extending algorithm.
2. Parallel implementation: From an implementation point of view parallelizability is also an important aspect of any domain extending algorithm.

Bellare and Rogaway [1] suggested the XOR tree hash (XTH) construction in order to reduce the key length expansion. Since XTH is based on the complete (or full) l -ary tree ($l \geq 2$), it has also an efficiency regarding the parallelizability (the processing speed). XTH had been the most efficient construction not only regarding the key length expansion but also regarding the parallelizability before Shoup's construction was presented in [10]. Shoup's construction is more efficient than XTH with regard to the key length expansion. Furthermore, Mironov [4] had shown that the key length expansion needed in Shoup's construction is the minimum possible for any sequential algorithm. In other words, there is no sequential algorithm which has more efficient key length expansion than Shoup's. But his construction is not more efficient than XTH with regard to the parallelizability since it is based on the unary tree. In the following, ' $B < A$ ' means that A is more efficient than B regarding the key length expansion or parallelizability.

Key length expansion:	XTH < Shoup
Parallelizability:	Shoup < XTH

Sarkar's work [8] was an attempt to propose a parallel algorithm which has the following properties:

- The algorithm's key length expansion is as good as possible.
- The algorithm's parallelizable efficiency is the same as XTH.

Therefore, he also chose the complete tree to obtain the same parallelizable performance as XTH and chose binary structure to adopt both of the mask assignment methods of Shoup's and XTH algorithm so that the key length expansion can be reduced as much as possible. As a result, Sarkar's construction has the same parallelizable performance as XTH. However, his construction does not have the same key length expansion as Shoup's one.

Key length expansion:	XTH < Sarkar < Shoup
Parallelizability:	Shoup < XTH = Sarkar

In this paper we will first present a tree based domain extension whose key length expansion is significantly less than Sarkar's construction. Furthermore, its parallelizable efficiency is the same as Sarkar's since it is also based on the complete binary tree. It will be called Improved Binary Tree based Construction

(IBTC). In fact, we have got a lot of evidences in [6] that IBTC will be optimal in the class of complete binary tree based algorithm. But only disadvantage is that here we need more masks (part of the key) than sequential construction. But it is not too large as in all practical situations we need just two more masks than Shoup's construction.

However, note that all the previously proposed parallel algorithms, including our first new construction, took more key length expansion than that of Shoup's sequential algorithm. So an important question is whether this is true in general of any parallel algorithm. Our second new construction shows that this is not the case.

The following is our motivation to design the second new parallel algorithm. At the present stage, it seems that the parallel constructions based on the complete l -ary tree have the most efficient parallelizability. But we think it is difficult to construct the parallel domain extender which has the same key length expansion as Shoup's sequential domain extender if we can only use the complete l -ary tree. Therefore, we decide to take somewhat different approach as follows without the assumption that we can use only the complete l -ary tree: In contrast to [8] and our first construction, this work is an attempt to propose a parallel algorithm which has the following properties:

- The algorithm has the same key length expansion as Shoup's.
- The algorithm's parallelizable efficiency is as good as possible.

As a result, the second new construction has the same key length expansion as Shoup's one. But the construction does not have the same parallelizable performance with our first new construction. The construction will be called l -DIMENSIONAL construction (l -DIM, $l \geq 2$).

Key length expansion:	XTH < Sarkar < IBTC < Shoup = l-DIM
Parallelizability:	Shoup < l-DIM < XTH = Sarkar = IBTC

The results may be summarized as shown in Table 1 (Here, 'seq' means 'sequential' and 'par' means 'parallel'). A more detailed comparison is presented in Table 2 in Section 7.

Table 1. Comparison of domain extenders for UOWHF

Method	Used Tree	Ranking of Key expansion	Ranking of Parallelizability	Seq /Par
BLH [1]	Unary	7	3	seq
XLH [1]	Unary	6	3	seq
Shoup [10]	Unary	1	3	seq
BTH [1]	Complete l -ary ($l \geq 2$)	5	1	par
XTH [1]	Complete l -ary ($l \geq 2$)	4	1	par
Sarkar [8]	Complete Binary	3	1	par
IBTC (this paper)	Complete Binary	2	1	par
l-DIM (this paper)	Non-Complete l -ary ($l \geq 2$)	1	2	par

We think it is difficult to say that which one is more important than the other between the key length expansion and the parallel implementation. Of course, it would be very nice to have a regular parallel structure something like the complete tree which also minimizes the key length expansion. But at this point, we do not have any such algorithm and IBTC is the best known construction among the complete binary tree based constructions. Hence, in our opinion, we should separately consider both the above-mentioned two points of view with the same importance. And the present works are important in regarding the former and the latter point of views, respectively. Particularly, the l -DIM and Shoup's one are the only two known algorithms which minimize the key length expansion. In addition that, the reason why the l -DIM has more meaning is that it is a parallel algorithm which has the same key length expansion as Shoup's sequential algorithm and this is the very first trial in designing the parallel algorithms.

Using the recent result [9], we can also prove that the key length expansion of our new parallel construction and Shoup's sequential construction are the minimum possible for any constructions in a large class of "natural" domain extenders including all the previously proposed methods.

We also give a sufficient condition for valid domain extension for sequential construction and it is likely that the condition is necessary. So, that will characterize the valid domain extension for sequential construction. In [6] M. Nandi has also shown that the same condition becomes sufficient for general tree based domain extension.

Related Work: Note that all of the above described parallel constructions are based on the assumption that the number of processors grows with the length of the message. In [9], Sarkar has first suggested a parallel domain extending algorithm which can be implemented with finitely many processors. But it does not have the same key length expansion as Shoup's. Here, it should be noted that his work mainly focuses on the parallel implementation with finite processors, on the contrary, the present work focuses on the parallel implementation with optimal key length expansion. And it seems that using the technique of [9], our new parallel constructions can be modified to the constructions which can work with finite processors.

3 Preliminaries

The following notations are used in this paper.

1. $[a, b] = \{a, a + 1, \dots, b\}$ where a and b are integers.
2. Suppose A is a finite set. By $a \in_R A$ we mean that a is a uniform discrete random variable taking values from A .
3. $\nu_2(i) = j$ if $2^j | i$ and $2^{j+1} \nmid i$.
4. For $t > 1$, $\log_2^m(t)$ means that the function \log_2 applies m many times on t . $\log_2^*(t) = m$ if $\log_2^m(t) \leq 1$ but $\log_2^{m-1}(t) > 1$.

5. In the complete binary tree based construction, $T_t = (V_t, E_t)$ means the complete binary tree where $V_t = \{1, 2, \dots, 2^t - 1\}$ is a node set and $E_t = \{e_i : 2 \leq i \leq 2^t - 1\}$ is a directed edge set where $e_i = (i, \lfloor i/2 \rfloor)$. Here $e_i = (v, w)$ denotes a directed edge, i.e., v is the initial node and w the terminal node. $ht_t(i) = j$ means that $2^{t-j} \leq i \leq 2^{t+1-j} - 1$. So, the root node has height t and all leaves have height 1. For any node i , define $T_t[i]$ by the complete binary sub-tree rooted at i .
6. In the 4-dimensional construction, for integer t , $g(t) = (a, b, c, d)$, where $a = \lfloor t/4 \rfloor + \lfloor ((t \bmod 4) + 3)/4 \rfloor$, $b = \lfloor t/4 \rfloor + \lfloor ((t \bmod 4) + 2)/4 \rfloor$, $c = \lfloor t/4 \rfloor + \lfloor ((t \bmod 4) + 1)/4 \rfloor$, and $d = \lfloor t/4 \rfloor$. Here $t \bmod 4 = t - \lfloor t/4 \rfloor \cdot 4$.
7. In the 4-dimensional construction let $T_t = (V_t, E_t)$ be a non-complete 4-ary tree, where $V_t = \{1, 2, \dots, 2^t\}$ and $E_t = \{e_i : 2 \leq i \leq 2^t\}$ where $e_i = (i, i-1)$ for $2 \leq i \leq 2^a$, $e_i = (i, i-2^a)$ for $2^a < i \leq 2^{a+b}$, $e_i = (i, i-2^{a+b})$ for $2^{a+b} < i \leq 2^{a+b+c}$, and $e_i = (i, i-2^{a+b+c})$ for $2^{a+b+c} < i \leq 2^t$. Here a, b, c , and d are such that $g(t) = (a, b, c, d)$.

Let $\{h_k\}_{k \in \mathcal{K}}$ be a keyed family of hash functions, where each $h_k : \{0, 1\}^n \rightarrow \{0, 1\}^m$, $n > m$. Consider the following adversarial game.

1. Adversary chooses an $x \in \{0, 1\}^n$.
2. Adversary is given a k which is chosen uniformly at random from \mathcal{K} .
3. Adversary has to find x' such that $x \neq x'$ but $h_k(x) = h_k(x')$.

A strategy \mathcal{A} for the adversary runs in two stages. In the first stage $\mathcal{A}^{\text{guess}}$, the adversary finds the x to which he has to commit in Step 1. It also produces some auxiliary state information σ . In the second stage $\mathcal{A}^{\text{find}}(k, x, \sigma)$, the adversary either finds a $x' \neq x$ such that $h_k(x) = h_k(x')$ or reports failure. Both $\mathcal{A}^{\text{guess}}$ and $\mathcal{A}^{\text{find}}(k, x, \sigma)$ are probabilistic algorithms. The success probability of the strategy is measured over the random choices made by $\mathcal{A}^{\text{guess}}$ and $\mathcal{A}^{\text{find}}(k, x, \sigma)$ and the random choice of k in Step 2 of the game.

We say that \mathcal{A} is an (ε, η) -strategy for $\{h_k\}_{k \in \mathcal{K}}$ if the success probability of \mathcal{A} is at least ε and it invokes the hash function h_k at most η times. In this case we say that the adversary has an (ε, η) -strategy for $\{h_k\}_{k \in \mathcal{K}}$. Note that we do not include time as an explicit parameter though it would be easy to do so. Informally, we say that $\{h_k\}_{k \in \mathcal{K}}$ is a UOWHF if the adversary has a negligible probability of success with respect to any probabilistic polynomial time strategy. Here, the security parameter is length of the message i.e., the length of the input.

In this paper we are interested in extending the domain of a UOWHF. More specifically, given a UOWHF $\{h_k\}_{k \in \mathcal{K}}$, $h_k : \{0, 1\}^n \rightarrow \{0, 1\}^m$, $n > m$, we would like to construct another extended UOWHF $\{H_p\}_{p \in \mathcal{P}}$ with $H_p : \{0, 1\}^N \rightarrow \{0, 1\}^m$, where $n < N$.

We say that \mathcal{B} is an (ε, η) -extended strategy for $\{H_p\}_{p \in \mathcal{P}}$ if the success probability of \mathcal{B} is at least ε and it invokes the hash function h_k at most η times. In this case we say that the adversary has an (ε, η) -extended strategy for $\{H_p\}_{p \in \mathcal{P}}$. Note that H_p is built using h_k and hence while studying strategies for H_p we are interested in the number of invocations of the hash function h_k .

The correctness of our construction will essentially be a Turing reduction. We will show that if there is an (ε, η) -extended strategy \mathcal{B} for $\{H_p\}_{p \in \mathcal{P}}$, then there

is an (ε', η') -strategy \mathcal{A} for $\{h_k\}_{k \in \mathcal{K}}$, where ε' is not significantly lesser than ε and η' is not much larger than η . This shows that if $\{h_k\}_{k \in \mathcal{K}}$ is a UOWHF, then so is $\{H_p\}_{p \in \mathcal{P}}$. In this case, we say that the domain extension is valid.

The key length for the base hash family $\{h_k\}_{k \in \mathcal{K}}$ is $\lceil \log_2 |\mathcal{K}| \rceil$. On the other hand, the key length for the extended hash family $\{H_p\}_{p \in \mathcal{P}}$ is $\lceil \log_2 |\mathcal{P}| \rceil$. Thus increasing the size of the input from n bits to N bits results in an increase of the key size by an amount $\lceil \log_2 |\mathcal{P}| \rceil - \lceil \log_2 |\mathcal{K}| \rceil$. From a practical point of view it is very important to minimize this increase in the key length.

For the remainder of this paper we assume the following conventions.

1. $\{h_k\}_{k \in \mathcal{K}}$ is always the base hash family, where $\mathcal{K} = \{0, 1\}^K$ and $h_k : \{0, 1\}^n \rightarrow \{0, 1\}^m$. In case of sequential construction $n > m$, in case of full binary tree based construction $n > 2m$, and in case of 4-dimensional construction $n > 4m$.
2. We will construct $\{H_p\}_{p \in \mathcal{P}}, H_p : \{0, 1\}^N \rightarrow \{0, 1\}^m$ using the base hash family $\{h_k\}_{k \in \mathcal{K}}$, where $p = k || \mu_1 || \mu_2 || \dots || \mu_l$ for some l and each μ_i is m -bit binary string called *mask* and $|k| = K$. Here, in case of sequential algorithm $N = n(r+1) - mr$, in case of tree based construction $N = n(2^t - 1) - m(2^t - 2)$ and in case of 4-dimensional construction $N = n2^t - m(2^t - 1)$. Let us define $\mu[i, j] = \mu_i || \dots || \mu_j$, where $1 \leq i \leq j \leq l$. We will use $\mu[j]$ instead of $\mu[1, j]$ for $j \geq 1$ and define $\mu[0]$ to be empty string.
3. In sequential construction input of H_p is written as $y = y_0 || y_1 || \dots || y_r$ where $|y_0| = n$ and $|y_i| = n - m$ for $1 \leq i \leq r$. In case of tree based construction input of H_p is written as $x = x_1 || \dots || x_{2^t-1}$ where $|x_i| = n - 2m$ for $1 \leq i < 2^{t-1}$ and $|x_i| = n$ for $2^{t-1} \leq i \leq 2^t - 1$. In 4-dimensional construction input of H_p is written as $x = x_1 || \dots || x_{2^t}$ where $|x_i| = n - 4m$ for $1 \leq i < 2^a$, $|x_i| = n - 3m$ for $2^a \leq i \leq 2^a(2^b - 1)$, $|x_i| = n - 2m$ for $2^a(2^b - 1) < i \leq 2^{a+b}(2^c - 1)$, $|x_i| = n - m$ for $2^{a+b}(2^c - 1) < i \leq 2^{a+b+c}(2^d - 1)$, and $|x_i| = n$ for $2^{a+b+c}(2^d - 1) + 1 \leq i \leq 2^t$. Here a, b, c , and d are such that $g(t) = (a, b, c, d)$.
4. In tree based construction let $i \in V_t$ and x be a message of length N . We define $x(i) = x_i || x_{2i} || x_{2i+1} || x_{4i} || x_{4i+1} || \dots$ i.e. concatenating all x_j in ascending order of j where j runs in $T_t[i]$. In other words the part of the message used in the complete binary sub-tree rooted at i .

4 Sequential Construction

The best known sequential algorithm is given by Shoup [10]. We will generalize the idea of the construction. We also give the sufficient condition for valid sequential construction. Let $\psi : [1, r] \rightarrow [1, l]$ be any function called a **masking assignment**. Fix a masking assignment ψ , $H_p(y)$, the extended hash function, is computed by the following algorithm.

1. **Input:** $y = y_0 || y_1 || \dots || y_r$ and $p = k || \mu_1 || \mu_2 || \dots || \mu_l$.
2. $z_0 = h_k(y_0)$.
3. For $1 \leq i \leq r$, define $s_i = z_{i-1} \oplus \mu_{\psi(i)}$ and $z_i = h_k(s_i || y_i)$.
4. **Output:** z_r .

We say that the sequential construction is based on the masking assignment ψ . In Shoup's algorithm $\psi = \nu_2 + 1$ and $l = 1 + \lfloor \log_2 r \rfloor$ (in his paper ν_2 is masking assignment but that makes no difference). We will write $s(i, y, k, \mu)$, $z(i, y, k, \mu)$ for s_i and z_i respectively (in the algorithm with input (y, p) , where $p = k \parallel \mu$). Now we will define some terms related with masking assignment and domain extension.

Definition 1. We say that ψ is **correct** if for all $1 \leq i \leq r$, $C \in \{0, 1\}^m$, $y \in \{0, 1\}^N$ and for any hash function h_k there is an algorithm called $Mdef_{seq}(i, y, k, C, \psi)$ which outputs $\mu = \mu_1 \parallel \mu_2 \parallel \dots \parallel \mu_l$ such that $s(i, y, k, \mu) = C$. $Mdef_{seq}(i, y, k, C, \psi)$ is called a mask defining algorithm. A sequential construction based on a correct masking assignment is called a correct domain extension. A masking assignment is **totally correct** if there is a mask defining algorithm $Mdef_{seq}(i, y, k, C, \psi) = \mu = \mu_1 \parallel \mu_2 \parallel \dots \parallel \mu_l$ for any i, y, k, C as above such that $s(i, y, k, \mu) = C$ holds and μ is a random string whenever C is a random string and other inputs are fixed.

Definition 2. We say that a domain extension algorithm is **valid** if $\{H_p\}_{p \in \mathcal{P}}$ is a UOWHF whenever $\{h_k\}_{k \in \mathcal{K}}$ is a UOWHF. In case of sequential construction if valid domain extension algorithm is based on a masking assignment ψ then we say that the masking assignment is **valid**.

Definition 3. A masking assignment $\psi : [1, r] \rightarrow [1, l]$ is **strongly even-free** (or **even-free**) if for each $[a, b] \subseteq [1, r]$ there exists $c \in [a, b]$ such that $\psi(c)$ occurs exactly once (respectively, odd times) in the sequence $\psi(a), \psi(a+1), \dots, \psi(b)$. Call this c (also the mask $\psi(c)$) a **single-man** for the interval $[a, b]$.

Now we will try to characterize all valid masking assignments. From Mironov's paper [4] we have seen that every valid masking assignment is even-free. He also showed that, every even-free masking assignment requires at least $1 + \lfloor \log_2 r \rfloor$ many masks and the minimum attains if we consider the masking assignment $\psi = \nu_2 + 1$ which is used in Shoup's algorithm. Now we will prove that, in case of sequential construction, every strongly even-free masking assignment is valid. The same masking assignment i.e. $\nu_2 + 1$ is in fact a strongly even-free masking assignment.

To provide the sufficient condition for valid sequential extension, we will first prove that strongly even-free implies totally correct. The proof of totally correct implies valid is a basic idea of proving an extension is valid. In all known papers the same idea is used for proving validness of extension. So, one can see this any one of these papers [8, 10, 4]. We will give a proof in case of complete binary tree based domain extension in Section 5.

Lemma 1. If ψ is strongly even-free then ψ is totally correct.

Proof. We will define the mask defining algorithm $Mdef_{seq}(i, y, k, C, \psi)$.

1. If $i = 1$ then define $\mu_{\psi(1)} = C \oplus h_k(y_0)$ and define all yet undefined masks randomly and quit.

2. If $i > 1$ then choose any c which is a single-man for the interval $[1, i]$. Compute $j \leftarrow i - c$. If $j = 0$ then goto step 4.
3. Let $\psi' : [1, j] \rightarrow [1, l]$ be a masking assignment such that $\psi'(n) = \psi(n + c)$ where $n \in [1, j]$. Take a random string D and then define, $y' = y'_0 || \dots || y'_j$ where, $y'_n = y_{n+c}$ when $n \geq 1$ and $y'_0 = D || y_c$. Run $\text{Mdef}_{seq}(j, y', k, C, \psi')$.
4. Define all yet undefined masks except $\mu_{\psi(c)}$ (i.e. after running Mdef_{seq} some masks may not be defined as ψ' may not be onto or j can be 0) randomly. Compute $\mu_{\psi(c)} = z(c - 1, y, k, \mu) \oplus D$ and quit.

Note that to compute $z(c - 1, y, k, \mu)$ we do not need the mask $\mu_{\psi(c)}$ as c is a single-man and the above recursive algorithm will always stop as $j < i$. The masking assignment ψ' is nothing but ψ restricted at $[c, i]$. So, if $s(c, y, k, \mu) = D$ then by induction $s(i, y, k, \mu) = C$. But, $s(c, y, k, \mu) = D$ is true by definition of $\mu_{\psi(c)}$. It proves the correctness of ψ . If C is a random string then all masks μ is a random string as they are randomly defined (in step-4) or they are obtained by XOR-ing with a random string (in step-1). So, it is totally correct. ■

Theorem 1. (Sufficient Condition for Valid Sequential Extension)

If a sequential domain extension is based on a strongly even-free masking assignment ψ then the domain extension is valid.

Proof. By the above lemma ψ is totally correct. The proof of totally correct implies valid is given in case of complete binary tree domain extension in Section 5. The same idea will carry through in case of sequential construction. So, we omit this proof.

Remark: Strongly even-free is sufficient condition for correct masking assignment. For example $\nu_2 + 1$. One can feel that the condition may be necessary. So, we may conjecture that, if a masking assignment is correct for any arbitrary hash function then it should be strongly even-free.

5 Complete Binary Tree Based Construction

In the previous section we study about sequential construction. Now, we will first define the generic algorithm based on complete binary tree of height t . Let $T_t = (V_t, E_t)$ be the full binary tree where $V_t = \{1, 2, \dots, 2^t - 1\}$ and $E_t = \{e_i; 2 \leq i \leq 2^t - 1\}$, $e_i = (i, \lfloor i/2 \rfloor)$. Let any function $\psi_t : E_t \rightarrow [1, l]$ be a masking assignment. (Note that we use E_t for domain of ψ_t .) Let $x = x_1 || \dots || x_{2^t-1}$ be the input message of length N . Given ψ_t, x , and $p = k || \mu$, $H_p(x)$ is computed by the following algorithm.

1. **Input:** $x = x_1 || x_2 || \dots || x_{2^t-1}$ and $p = k || \mu_1 || \mu_2 || \dots || \mu_l$.
2. If $2^{t-1} \leq i \leq 2^t - 1$ then $z_i = h_k(x_i)$ else if $1 \leq i < 2^{t-1}$ then $z_i = h_k(s_{2i} || s_{2i+1} || x_i)$, where $s_i = z_i \oplus \mu_{\psi_t(e_i)}$.
3. **Output:** z_1 .

Note that the input of i^{th} node is $s_{2i} || s_{2i+1} || x_i$ and output of node i is z_i . We say that the above complete binary tree based domain extension is based on

the masking assignment ψ_t . We will write $s(i, x, k, \mu, t)$ and $z(i, x, k, \mu, t)$ for s_i and z_i in the above algorithm, respectively. Like sequential algorithm we say that ψ_t is **correct** if for each $1 \leq i < 2^{t-1}$, there is a mask defining algorithm $\text{Mdef}_{\text{tree}}(i, x, k, t, r_0, r_1, \psi_t)$ where $|r_0| = |r_1| = m$ which outputs $\mu = \mu_1 || \dots || \mu_l$ such that $s(2i, x, k, \mu, t) = r_0$ and $s(2i+1, x, k, \mu, t) = r_1$. ψ_t is **totally correct** if the output μ of the mask defining algorithm is random string provided r_0, r_1 are random strings and other inputs are fixed.

Definition 4. A masking assignment $\psi_t : E_t \rightarrow [1, l]$ is a **level uniform masking assignment** if there are two functions $\alpha_t, \beta_t : [2, t] \rightarrow [1, l]$ such that $\psi_t(e_i) = \alpha_t(j)$ if i is odd and $\psi_t(e_i) = \beta_t(j)$ if i is even, where $j = h_{t-1}(i) + 1$.

We will first briefly state some standard binary tree based constructions all of which are based on level-uniform masking assignment.

1. **Bellare-Rogaway** [1]: $\alpha_t(i) = i - 1$ and $\beta_t(i) = t + i - 2$. In [1] it was shown that ψ_t is valid. Here, we need $2(t - 1)$ masks.
2. **Sarkar** [8]: $\alpha_t(i) = i - 1$ and $\beta_t(i) = t + \nu_2(i - 1)$. In [8] it was shown that ψ_t is valid. Here, we need $t + \lceil \log_2 t \rceil - 1$ masks.

Now, we will propose our binary tree based construction which needs lesser number of masks than Sarkar's. Like above examples our domain extension is also based on level uniform masking assignment. So, it is enough to define these two functions α_t and β_t . This construction can be found more detail in [5].

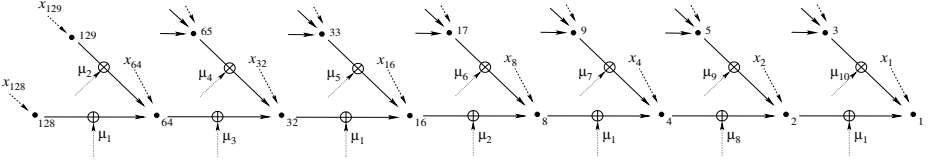


Fig. 1. The right most part of the complete binary tree when you place the root of the tree (i.e. vertex 1) in top. ($t = 8$ and $|x_1| = \dots = |x_{127}| = n - 2m$, and $|x_{128}| = \dots = |x_{255}| = n$. \bullet means $h_k(\cdot)$.)

5.1 Improved Binary Tree Based Construction

Define two sequences $\{l_k\}_{k \geq 0}$ and $\{m_t\}_{t \geq 2}$ as follow: $l_{k+1} = 2^{l_k+k} + l_k$ where, $l_0 = 2$ and $m_2 = 2$ and if $k \geq 1$, $m_t = t + k$ for all $t \in [l_{k-1} + 1, l_k]$. Note that, both l_k and m_t are strictly increasing sequences and if $t = l_k$ for some k then $m_{t+1} = m_t + 2$ and if for some k , $l_k < t < l_{k+1}$ then $m_{t+1} = m_t + 1$. Later, we will see that m_t is the number of masks of our algorithm for binary tree of height t and $m_t \leq t + k$ till $t \leq l_k$. Intuitively, $k = O(\log_2^*(l_k))$ so, $m_t = t + O(\log_2^* t)$. The level uniform masking assignment ψ_t is based on the functions $\alpha_t, \beta_t : [2, t] \rightarrow [1, m_t]$, $t \geq 2$ where they are defined as follow (See Figure 1 where a right most part of the tree is drawn which will completely determine the functions α_t and β_t):

1. $\alpha_2(2) = 2$ and $\beta_2(2) = 1$.
2. For $t \geq 3$, $\alpha_t(i) = \alpha_{t-1}(i)$ and $\beta_t(i) = \beta_{t-1}(i)$ whenever $2 \leq i \leq t-1$.
3. If $t \geq 3$ and $t-1 = l_k$ for some k then $\alpha_t(t) = \alpha_{t-1}(t-1) + 2$ and $\beta_t(t) = \alpha_{t-1}(t-1) + 1$ and if $l_k < t-1 < l_{k+1}$ then $\alpha_t(t) = \alpha_{t-1}(t-1) + 1$ and $\beta_t(t) = \nu_2(t-1-l_k) + 1$.

Theorem 2. For $t \geq 2$, α_t and β_t map into $[1, m_t]$. Moreover, $\alpha_t(t) = m_t$ and $\alpha_t([2, t]) \cup \beta_t([2, t]) = [1, m_t]$. So, we need m_t many masks.

Proof. For $2 \leq i \leq t$, $\alpha_t(i) = m_i$ can be easily proved by induction. Also note that, when $i = l_k + 1$ for some k , then $\beta_t(i) = m_{i-1} + 1 < m_i$ and when $l_k + 1 < i \leq l_{k+1}$, $\beta_t(i) = \nu_2(i - l_k) + 1 \leq l_k + k = m_{l_k} < m_i$. So, it proves that α_t and β_t map into $[1, m_t]$. To prove the last part let $1 \leq j \leq m_t$. So we have some i so that $j = m_i$ or $j = m_{l_k} + 1$. If $j = m_i$ then $\alpha_t(i) = m_i = j$ otherwise $\beta_t(l_k + 1) = m_{l_k} + 1 = j$. So, we have that $\alpha_t([2, t]) \cup \beta_t([2, t]) = [1, m_t]$. ■

Now, we will prove that the above ψ_t is totally correct for all $t \geq 2$. For this we need to define $\mathbf{Mdef}_{tree}(i, x, k, t, r_0, r_1, \psi_t)$. We will define the mask defining algorithm for $i = 1$ otherwise we can consider the complete binary tree rooted at i (i.e. $T_t[i]$) and define $\mathbf{Mdef}_{tree}(i, x, k, t, r_0, r_1, \psi_t)$ by $\mathbf{Mdef}_{tree}(1, x', k, t', r_0, r_1, \psi')$ where, $t' = ht_t(i)$, $x' = x(i)$ i.e. the part of the message involved in the subtree $T_t[i]$ and ψ' is ψ_t restricted at $T_t[i]$ which is same as $\psi_{t'}$ (it can be checked easily as ψ_t is level uniform). So, we can assume that $i = 1$.

1. If $t = l_k + 1$ for some k then
 - (a) Define μ by random string.
 - (b) Compute $\mu_{m_{t-1}} = z(2, x, k, \mu, t) \oplus r_0$ and $\mu_{m_t} = z(3, x, k, \mu, t) \oplus r_1$. To compute $z(2, x, k, \mu, t)$ and $z(3, x, k, \mu, t)$ we actually need only $\mu[m_t - 2]$ as $\mu_{m_{t-1}}$ and μ_{m_t} appear only on edges e_2 and e_3 .
2. If $l_k + 1 < t \leq l_{k+1}$ for some k then
 - (a) Let the set $A = \{2^{i+1} + 1 : 0 \leq i \leq r\} \cup \{2^{r+1}\}$ where, $r = t - (l_k + 1)$.
 - (b) Choose b_i randomly for all $i \in A - \{3\}$ such that, $|b_i| = m$ and $b_3 = r_1$.
 - (c) Let $y = y_0 || y_1 || \dots || y_r$ where, $y_0 = b_{2^{r+1}} || b_{2^{r+1}+1} || x_{2^r}$ and $y_j = b_{2^{r+1}-j+1} || x_{2^{r-j}}$ for $1 \leq j \leq r$.
 - (d) Run $\mathbf{Mdef}_{seq}(r, y, k, r_0, \psi') = \mu[l']$ where, $l' = \lfloor \log_2 r \rfloor + 1 \leq l_k + k = m_{l_k}$ and ψ' is same as ψ restricted at the path $e_{2^r}, e_{2^{r-1}}, \dots, e_2$. More precisely, $\psi'(i) = \psi(e_{2^{r+1}-i})$. So, if μ is computed such a way that, $s(i, x, k, \mu, t) = b_i$ for all $i \in A$ then by definition of \mathbf{Mdef}_{seq} we will have $s(2, x, k, \mu, t) = r_0$.
 - (e) Define remaining masks randomly and for $i \in A$ (in descending order) compute $\mu_i = z(i, x, k, \mu, t) \oplus b_i$.

When $t = l_k + 1$ the correctness of ψ_t easily follows from the definition of $\mu_{m_{t-1}}$ and μ_{m_t} . Note that, to compute $z(j, x, k, \mu, t)$ for $j \in A$ in step-2(e), we do not need the masks $\mu_{\psi(e_{j'})}$ for all $j' \in A$, $j' > j$. So, $s(i, x, k, \mu, t) = b_i$ for all $i \in A$ and hence \mathbf{Mdef}_{tree} is correct. If r_0 and r_1 are random strings then so is the output μ and hence ψ_t is totally correct for all t . So, we have the following theorem:

Theorem 3. *The masking assignment ψ_t based on two functions α_t and β_t as above is totally correct.*

Now we will prove the statement *totally correct implies valid* for binary tree based masking assignment. The same idea will carry through for the other constructions.

Theorem 4. (Validness of domain extension) *In case of binary tree based domain extension a totally correct masking assignment is always valid. More precisely, we have that, if there is an (ε, η) winning strategy \mathcal{A} for $\{H_p\}_{p \in \mathcal{P}}$ then there is also an $(\frac{\varepsilon}{2^t-1}, \eta + 2(2^t - 1))$ -strategy \mathcal{B} for $\{h_k\}_{k \in \mathcal{K}}$ whenever $\{H_p\}_{p \in \mathcal{P}}$ is based on totally correct masking assignment.*

Proof. We describe the two stages of the strategy \mathcal{B} as follows.

Algorithm $\mathcal{B}^{\text{guess}} = (y, s)$:

Run $\mathcal{A}^{\text{guess}}$ to obtain $x \in \{0, 1\}^N$ and state information s' . Choose an $i \in_R \{1, \dots, 2^t - 1\}$. If $2^{t-1} \leq i \leq 2^t - 1$, set $y = x_i$; r_0, r_1 to be the empty string and $s = (s', i, r_0, r_1, x)$. Output (y, s) and stop. If $1 \leq i \leq 2^{t-1} - 1$, then choose two strings r_0 and r_1 uniformly at random from the set $\{0, 1\}^m$. Set $y = r_0 || r_1 || x_i$ and $s = (s', i, r_0, r_1, x)$. Output (y, s) and stop. At this point the adversary is given a k which is chosen uniformly at random from the set $\mathcal{K} = \{0, 1\}^K$. The adversary then runs $\mathcal{B}^{\text{find}}$ which is described below.

Algorithm $\mathcal{B}^{\text{find}}(y, k, s) = y'$: (Note $s = (s', i, r_0, r_1, x)$.)

Define the masks μ_1, \dots, μ_{m_t} by executing algorithm $\text{Mdef}_{\text{tree}}(i, x, k, t, r_0, r_1)$. This defines the key $p = k || \mu$ for the function H_p . Run $\mathcal{A}^{\text{find}}(x, p, s')$ to obtain x' . Let y' be the input of i^{th} node corresponding to the string x' . Output y' .

We now lower bound the probability of success. By total correctness p is a randomly chosen key from the set \mathcal{P} . Suppose x and x' ($x \neq x'$) collide for the function H_p . Then there must be a j in the range $1 \leq j \leq 2^t - 1$ such that at vertex j there is a collision for the function h_k . (Otherwise it is possible to prove by a backward induction that $x = x'$.) The probability that $j = i$ is $\frac{1}{2^t-1}$ where i is a random number lying between 1 and $2^t - 1$. Hence if the success probability of \mathcal{A} is at least ε , then the success probability of \mathcal{B} is at least $\frac{\varepsilon}{2^t-1}$. Also the number of invocations of h_k by \mathcal{B} is equal to the number of invocations of h_k by \mathcal{A} plus at most $2(2^t - 1)$. This completes the proof. ■

Theorem 5. *The speed-up of our algorithm over the sequential algorithm in Section 4 is by a factor of $\frac{2^t-1}{t}$.*

Proof. This algorithm hashes a message of length $n(2^t - 1) - m(2^t - 2)$ into a digest of length m using t parallel rounds. The time taken by a single parallel round is proportional to the time required by a single invocation of the hash function h_k . The sequential construction require $2^t - 1$ invocations of the hash function h_k on a message of length $n(2^t - 1) - m(2^t - 2)$. Hence, the speed-up of the binary tree algorithm over the sequential algorithm is by a factor of $\frac{2^t-1}{t}$. ■

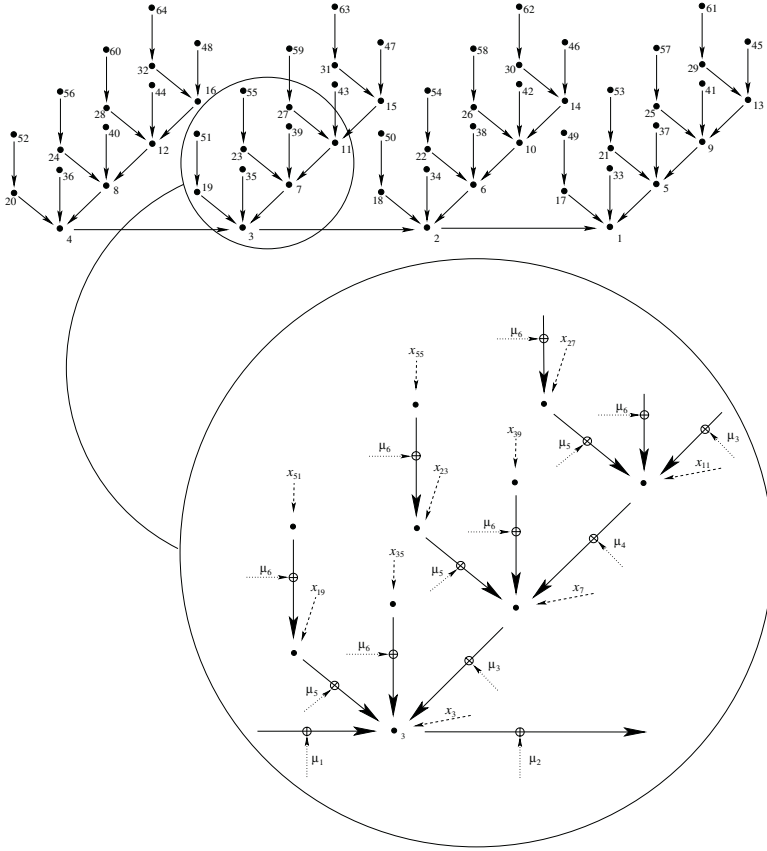


Fig. 2. 4-dimensional parallel algorithm ($t = 6$ and $x = x_1 || \dots || x_{26}$. Note that $g(6) = (2, 2, 1, 1)$ and $|x_1| = \dots = |x_3| = n - 4m$, $|x_4| = \dots = |x_{12}| = n - 3m$, $|x_{13}| = \dots = |x_{16}| = n - 2m$, $|x_{17}| = \dots = |x_{32}| = n - m$, and $|x_{33}| = \dots = |x_{26}| = n$. \bullet means $h_k(\cdot)$.)

Remark: The speed-up achieved by our algorithm is substantial even for moderate values of t . Such speed-up will prove to be advantageous for hashing long messages.

Theorem 6. *The number of masks for this algorithm is $t + O(\log_2^* t)$.*

Proof. From the recurrence relation it is clear that $2^{2^{l_k}} > l_{k+1} > 2^{l_k}$. So, $\log_2^*(l_k) + 1 \leq \log_2^*(l_{k+1}) \leq \log_2^*(l_k) + 2$ and hence $\log_2^*(l_k) = \theta(k)$ i.e. $\log_2^*(l_k)$ and k are of same order. So, for all $l_k \leq t < l_{k+1}$, $m_t - t = k = O(\log_2^* t)$. ■

6 Non-complete l -Ary Tree Based Construction

Our first new construction IBTC is based on the complete binary tree. In this section we present a new parallel construction for a UOWHF based on a 4-ary directed tree which is not complete.

We will first define the generic algorithm based on the 4-ary directed tree $T_t = (V_t, E_t)$ for $t \geq 4$ (See this notation in Section 3). For $t = 2$ and $t = 3$, we can define the algorithm based on the binary and 3-ary tree based construction (See Section 6.3), respectively.

Like previous constructions, any function $\psi_t : E_t \rightarrow [1, l]$ is a masking assignment. Let $x = x_1 || x_2 || \dots || x_{2^t}$ be the input message of length N . Given ψ_t, x , and $p = k || \mu, H_p(x)$ is computed by the following algorithm. This is depicted in Figure 2. In this section a, b, c and d denote the output of $g(t)$.

1. **Input:** $x = x_1 || x_2 || \dots || x_{2^t}$ and $p = k || \mu_1 || \mu_2 || \dots || \mu_l$.
2. If $2^{a+b+c}(2^d - 1) < i \leq 2^t$ then $z_i = h_k(x_i)$.
3. If $d = 1$ then goto step 4.
 - (a) For $j = 2^d - 2$ down to 1 do
 - For $j2^{a+b+c} < i \leq (j+1)2^{a+b+c}$, $z_i = h_k(s_{i+2^{a+b+c}} || x_i)$ where $s_i = z_i \oplus \mu_{\psi_t(e_i)}$ (This notation is also same in the following procedure).
4. For $2^{a+b}(2^c - 1) < i \leq 2^{a+b+c}$, $z_i = h_k(s_{i+2^{a+b+c}} || x_i)$.
5. If $c = 1$ go to step 6.
 - (a) For $j = 2^c - 2$ down to 1 do
 - For $j2^{a+b} < i \leq (j+1)2^{a+b}$, $z_i = h_k(s_{i+2^{a+b}} || s_{i+2^{a+b+c}} || x_i)$.
6. For $2^a(2^b - 1) < i \leq 2^{a+b}$, $z_i = h_k(s_{i+2^{a+b}} || s_{i+2^{a+b+c}} || x_i)$.
7. If $b = 1$ go to step 8.
 - (a) For $j = 2^b - 2$ down to 1 do
 - For $j2^a < i \leq (j+1)2^a$, $z_i = h_k(s_{i+2^a} || s_{i+2^{a+b}} || s_{i+2^{a+b+c}} || x_i)$.
8. For $i = 2^a$, $z_i = h_k(s_{i+2^a} || s_{i+2^{a+b}} || s_{i+2^{a+b+c}} || x_i)$.
9. For $i = 2^a - 1$ down to 1, $z_i = h_k(s_{i+1} || s_{i+2^a} || s_{i+2^{a+b}} || s_{i+2^{a+b+c}} || x_i)$.
10. **Output:** z_1 .

We say that, the above non-complete 4-ary tree based construction is based on the masking assignment ψ_t . Here, we need some definitions in order to consider the correctness of ψ_t .

1. We will write $s(i, x, k, \mu, t)$, $z(i, x, k, \mu, t)$ for s_i and z_i , respectively.
2. ϵ means the empty string.
3. For each node $1 \leq i \leq 2^{a+b+c}(2^d - 1)$,
 - (a) Define $s^0(i, x, k, \mu, t)$ as $s(i+1, x, k, \mu, t)$ for $1 \leq i < 2^a$ and as ϵ for $2^a \leq i \leq 2^{a+b+c}(2^d - 1)$.
 - (b) Define $s^1(i, x, k, \mu, t)$ as $s(i+2^a, x, k, \mu, t)$ for $1 \leq i \leq 2^a(2^b - 1)$ and as ϵ for $2^a(2^b - 1) < i \leq 2^{a+b+c}(2^d - 1)$.
 - (c) Define $s^2(i, x, k, \mu, t)$ as $s(i+2^{a+b}, x, k, \mu, t)$ for $1 \leq i \leq 2^{a+b}(2^c - 1)$ and as ϵ for $2^{a+b}(2^c - 1) < i \leq 2^{a+b+c}(2^d - 1)$.
 - (d) Define $s^3(i, x, k, \mu, t)$ as $s(i+2^{a+b+c}, x, k, \mu, t)$ for $1 \leq i \leq 2^{a+b+c}(2^d - 1)$.

Therefore the input of i^{th} node can be represented by $s^0(i, x, k, \mu, t) || s^1(i, x, k, \mu, t) || s^2(i, x, k, \mu, t) || s^3(i, x, k, \mu, t) || x_i$ for $1 \leq i \leq 2^{a+b+c}(2^d - 1)$.

We will say that ψ_t is **correct** if, for each $1 \leq i \leq 2^{a+b+c}(2^d - 1)$, there is an algorithm $\mathbf{Mdef}_{4dim}(i, x, k, t, r_0, r_1, r_2, r_3, \psi_t)$, where r_0 is a m -bit string if $1 \leq i < 2^a$ and ϵ if $2^a \leq i \leq 2^a(2^b - 1)$, r_1 is a m -bit string if $1 \leq i \leq 2^a(2^b - 1)$

and ϵ if $2^a(2^b - 1) < i \leq 2^{a+b}(2^c - 1)$, r_2 is a m -bit string if $1 \leq i \leq 2^{a+b}(2^c - 1)$ and ϵ if $2^{a+b}(2^c - 1) < i \leq 2^{a+b+c}(2^d - 1)$, and r_3 is a m -bit string for $1 \leq i \leq 2^{a+b+c}(2^d - 1)$ which outputs $\mu = \mu_1 || \dots || \mu_l$ such that $s^j(i, x, k, \mu, t) = r_j$ for $0 \leq j \leq 3$. ψ_t is **totally correct** if the output μ of the mask defining algorithm is random string provided r_0, r_1, r_2 and r_3 are random strings and other inputs are fixed.

6.1 4-Dimensional Domain Extender

Our second new parallel construction uses the following masking assignment $\psi_t : E_t \rightarrow [1, t]$. The map represents the assignment of masks to the directed edges. Here we present our definition of ψ_t which needs t masks for 4-dimensional construction. Intuitively, the map ψ_t is made from expanding the mask assigning method of Shoup's sequential construction into four directions. At first, we define four functions $\alpha_t, \beta_t, \gamma_t$, and δ_t as follows.

1. $\alpha_t : [1, 2^a - 1] \rightarrow [1, a]$ is defined by $\alpha_t(i) = 1 + \nu_2(2^a - i)$.
2. $\beta_t : [1, 2^b - 1] \rightarrow [a + 1, a + b]$ is defined by $\beta_t(i) = a + 1 + \nu_2(2^b - i)$.
3. $\gamma_t : [1, 2^c - 1] \rightarrow [a + b + 1, a + b + c]$ is defined by $\gamma_t(i) = a + b + 1 + \nu_2(2^c - i)$.
4. $\delta_t : [1, 2^d - 1] \rightarrow [a + b + c + 1, t]$ is defined by $\delta_t(i) = a + b + c + 1 + \nu_2(2^d - i)$.

Our masking assignment $\psi_t(e_i)$ is defined as follow:

1. $\psi_t(e_i) = \alpha_t(j)$ if $2 \leq i \leq 2^a$ and $j = i - 1$.
2. $\psi_t(e_i) = \beta_t(j)$ if $2^a < i \leq 2^{a+b}$ and $j2^a < i \leq (j + 1)2^a$.
3. $\psi_t(e_i) = \gamma_t(j)$ if $2^{a+b} < i \leq 2^{a+b+c}$ and $j2^{a+b} < i \leq (j + 1)2^{a+b}$.
4. $\psi_t(e_i) = \delta_t(j)$ if $2^{a+b+c} < i \leq 2^t$ and $j2^{a+b+c} < i \leq (j + 1)2^{a+b+c}$.

Now we will prove that the above ψ_t is totally correct.

Theorem 7. *The masking assignment ψ_t based on four functions $\alpha_t, \beta_t, \gamma_t$ and δ_t as above is totally correct.*

Proof. We will define the mask defining algorithm Mdef_{4dim} .

Input: $k, x, i, r_0, r_1, r_2, r_3, \psi_t$

output: $\mu = \mu_1 || \dots || \mu_t$ such that $s^j(i, x, k, \mu, t) = r_j$ for $0 \leq j \leq 3$.

We can define Mdef_{4dim} for each case $j \in \{1, 2, 3, 4\}$ where

1. $1 \leq i < 2^a$.
2. $2^a \leq i \leq 2^a(2^b - 1)$.
3. $2^a(2^b - 1) < i \leq 2^{a+b}(2^c - 1)$.
4. $2^{a+b}(2^c - 1) < i \leq 2^{a+b+c}(2^d - 1)$.

But we will present the specific procedure of Mdef_{4dim} for only case 1 since the other cases are very similar and much simpler than case 1. Let $1 \leq i < 2^a$.

1. (a) Let $D = 2^d - 1$. Let $\psi' : [1, D] \rightarrow [1, t]$ be a masking assignment such that $\psi'(j) = \psi_t(e_{i+(D+1-j)2^{a+b+c}})$ where $j \in [1, D]$.
 (b) Let $y^3 = y_0^3 || y_1^3 || \dots || y_D^3$ where, $y_v^3 = x_{i+(D-v)2^{a+b+c}}$ for $0 \leq v \leq D-1$ and $y_D^3 = r_0 || r_1 || r_2 || x_i$. Note that $|y_0^3| = n$ and $|y_j^3| = n - m$ for $1 \leq j \leq D$.
 (c) Run $\text{Mdef}_{seq}(D, y^3, k, r_3, \psi')$ to get an output $\mu[a + b + c + 1, t]$.
 (d) Set $\mu = \mu[t] = \mu'[a + b + c] || \mu[a + b + c + 1, t]$, where $\mu'[a + b + c]$ is the $m(a + b + c)$ -bit zero string.
2. (a) Let $C = 2^c - 1$. Let $\psi'' : [1, C] \rightarrow [1, t]$ be a masking assignment such that $\psi''(j) = \psi_t(e_{i+(C+1-j)2^{a+b}})$ where $j \in [1, C]$.
 (b) Let $y^2 = y_0^2 || y_1^2 || \dots || y_C^2$, where $y_v^2 = s^3(i + (C - v)2^{a+b}, x, k, \mu, t) || x_{i+(C-v)2^{a+b}}$, for $0 \leq v \leq C-1$ and $y_C^2 = r_0 || r_1 || r_3 || x_i$.
 (c) Run $\text{Mdef}_{seq}(C, y^2, k, r_2, \psi'')$ to get an output $\mu[a + b + 1, a + b + c]$.
 (d) Set $\mu = \mu[t] = \mu'[a + b] || \mu[a + b + 1, a + b + c] || \mu[a + b + c + 1, t]$, where $\mu'[a + b]$ is the $m(a + b)$ -bit zero string.
3. (a) Let $B = 2^b - 1$. Let $\psi''' : [1, B] \rightarrow [1, t]$ be a masking assignment such that $\psi'''(j) = \psi_t(e_{i+(B+1-j)2^a})$ where $j \in [1, B]$.
 (b) Let $y^1 = y_0^1 || y_1^1 || \dots || y_B^1$, where $y_v^1 = s^2(i + (B - v)2^a, x, k, \mu, t) || s^3(i + (B - v)2^a, x, k, \mu, t) || x_{i+(B-v)2^a}$, for $0 \leq v \leq B-1$ and $y_B^1 = r_0 || r_2 || r_3 || x_i$.
 (c) Run $\text{Mdef}_{seq}(B, y^1, k, r_1, \psi''')$ to get an output $\mu[a + 1, a + b]$.
 (d) Set $\mu = \mu[t] = \mu'[a] || \mu[a + 1, a + b] || \mu[a + b + 1, a + b + c] || \mu[a + b + c + 1, t]$, where $\mu'[a]$ is the ma -bit zero string.
4. (a) Let $u = 2^a - i$ and $A = 2^a - 1$. Let $\psi'''' : [1, A] \rightarrow [1, t]$ be a masking assignment such that $\psi''''(j) = \psi_t(e_{A+2-j})$ where $j \in [1, A]$.
 (b) Let $y^0 = y_0^0 || y_1^0 || \dots || y_A^0$ where, $y_v^0 = s^1(A + 1 - v, x, k, \mu, t) || s^2(A + 1 - v, x, k, \mu, t) || s^3(A + 1 - v, x, k, \mu, t) || x_{A+1-v}$ for $0 \leq v \leq A-1$ and $y_A^0 = 1^{3m} || x_1$.
 (c) Run $\text{Mdef}_{seq}(u, y^0, k, r_0, \psi''')$ to get an output $\mu[a]$.
5. Output $\mu[t] = \mu[a] || \mu[a + 1, a + b] || \mu[a + b + 1, a + b + c] || \mu[a + b + c + 1, t]$.

It is easy to check that $s^j(i, x, k, \mu, t) = r_j$ for $0 \leq j \leq 3$. Therefore Mdef_{4dim} is correct for $1 \leq i \leq 2^a - 1$. If r_0, r_1, r_2 , and r_3 are random strings then so is the output μ and hence ψ_t is totally correct for $1 \leq i \leq 2^a - 1$. The other cases are very similar. So we omit the proof for these cases. \blacksquare

The following theorem shows that if $\{h_k\}_{k \in \mathcal{K}}$ is a UOWHF, then so is $\{H_p\}_{p \in \mathcal{P}}$. Using the fact that ψ_t is totally correct, we can prove this theorem in a much similar way in the proof of Theorem 4. So we omit this proof.

Theorem 8. (Validness of domain extension) *In case of 4-dimensional domain extension a totally correct masking assignment is always valid. More precisely, if there is an (ε, η) -extended strategy for $\{H_p\}_{p \in \mathcal{P}}$ then there is an $(\frac{\varepsilon}{2^t}, \eta + 2^{t+1})$ -strategy for $\{h_k\}_{k \in \mathcal{K}}$ whenever $\{H_p\}_{p \in \mathcal{P}}$ is based on a totally correct masking assignment.*

We now show the speed-up of 4-dimensional construction over the sequential construction. For the sake of simplicity we do not describe the case of $t \not\equiv 0 \pmod{4}$.

Theorem 9. *The speed-up of 4-dimensional construction over the sequential construction in Section 4 is by a factor of $\frac{2^t}{2^{2+t/4}-3}$ if $t \equiv 0 \pmod{4}$.*

Proof. 4-dimensional construction hashes a message of length $n2^t - m(2^t - 1)$ into a digest of length m using $2^a + 2^b + 2^c + 2^d - 3$ parallel rounds. Therefore, if $t \equiv 0 \pmod{4}$ then $4 \times 2^{t/4} - 3$ parallel rounds are need to hash a message of length $n2^t - m(2^t - 1)$. The time taken by a single parallel round is proportional to the time required by a single invocation of the hash function h_k . The sequential construction require 2^t invocations of the hash function h_k on a message of length $n2^t - m(2^t - 1)$. Hence, the speed-up of the 4-dimensional construction over the sequential construction is by a factor of $\frac{2^t}{2^{2+t/4}-3}$ if $t \equiv 0 \pmod{4}$. ■

By the definition of the masking assignment of 4-dimensional construction, the following theorem is clear.

Theorem 10. *The number of masks for 4-dimensional construction is t .*

6.2 Optimality of the 4-Dimensional Domain Extender

in [4] Mironov proved that among all the sequential algorithms Shoup’s algorithm reuses the masks as much as possible. This means that among all the sequential algorithms there is no algorithm which has a more smaller key expansion than Shoup’s algorithm.

As Mironov did in [4], we can also ask whether the masks can be re-used even more in the 4-dimensional domain extender. But, luckily, we can easily answer the question using the recent result of Sarkar [8]. Furthermore, using the result, we can prove that the key length expansion of the 4-dimensional domain extender is the minimum possible for any algorithms in a large class of “natural” domain extending algorithms including all the 4-dimensional type algorithms and all the previously proposed algorithms.

In [8] Sarkar provided a generic lower bound on the key length expansion required for securely extending the domain of a UOWHF. He first defined the large class \mathcal{A} of “natural” domain extending algorithms. Then he proved that for any $A \in \mathcal{A}$ such that A is correct for s invocations of h_k the number of masks required by A is at least $\lceil \log_2 s \rceil$. (Details can be found in section 4 of [8].) Note that Shoup’s algorithm is an element of the class \mathcal{A} . Therefore, it follows that Shoup’s algorithm is optimal for the class \mathcal{A} .

On the other hand the 4-dimensional domain extender is also an element of the class \mathcal{A} . And note that for 2^t invocations of h_k the 4-dimensional domain extender uses $t(= \lceil \log_2 2^t \rceil)$ masks to securely extend the domain of a UOWHF. Hence this shows that the 4-dimensional domain extender is also optimal for the class \mathcal{A} .

6.3 l -Dimensional Domain Extender

In the above we provided the 4-dimensional domain extender and considered the security and optimality of key length expansion. In fact the construction idea

Table 2. Specific comparison of domain extenders for UOWHF.

Parameter	Shoup [10]	l -DIM ($l \geq 2$)	IBTC	Sarkar [8]
seq/par	sequential	parallel	parallel	parallel
message length	$2^t n$ $-(2^t - 1)m$	$2^t n$ $-(2^t - 1)m$	$(2^t - 1)n$ $-(2^t - 2)m$	$(2^t - 1)n$ $-(2^t - 2)m$
# invocations of h_k	2^t	2^t	$2^t - 1$	$2^t - 1$
# masks	t	t	$t + O(\log_2^* t)$	$t + \lceil \log_2 t \rceil - 1$
# rounds	2^t	$l2^{t/l} - l + 1 (t \equiv 0 \pmod l)$	t	t
speed-up	1	$\frac{2^t}{l2^{t/l} - l + 1} (t \equiv 0 \pmod l)$	$\frac{2^t - 1}{t}$	$\frac{2^t - 1}{t}$

can be generalized to any l -dimensional domain extender ($l \geq 2$). If $n \geq lm$, we can define the l -dimensional domain extender. We can start to define the l -dimensional domain extender with setting the function $g(t) = (a_1, \dots, a_l)$ exactly in the similar way as we did for 4-dimensional. And the whole specification of l -dimensional domain extender can be similarly defined by using the description method of the 4-dimensional domain extender. We can also consider the security and optimality of the l -dimensional domain extender as in the case of 4-dimensional domain extender.

7 Comparison to Known Algorithms

In Table 2 we compare the specific performance of the different known algorithms with l -dimensional domain extender and Improved binary tree based construction. Note that the message length which can be handled varies with each of the known algorithms. For example, Shoup's and l -DIM can handle a $2^t n - (2^t - 1)m$ bits message, however, Sakar's and IBTC can not handle the same length message. Therefore, we can not fix a message length in order to compare the different known algorithms with l -DIM and IBTC. Instead, we separately describe the message length for each of the algorithms as shown in Table 2.

The algorithms use one key for the base hash function and some number of m -bit mask keys. The number of masks described in Table 2 refers to the latter. The number of invocations of h_k is the total cost. The number of rounds reflects the parallelizability arising via tree-based constructions, and indicates the total time to completion. In Shoup's sequential construction it is equal to the number of invocations of h_k . Speed-up (over the sequential algorithm or Shoup) is the ratio of the number of invocations of h_k to that of rounds. For the sake of simplicity we do not describe the case of $t \not\equiv 0 \pmod l$ in the positions of the number of rounds and speed for our l -DIM.

Table 2 shows the key length expansion of l -DIM is the same as that of Shoup's and it doesn't have the same parallelizable performance with IBTC and Sarkar's construction. But if l is getting larger, then the speed of the l -DIM is also getting near to the speed of IBTC and Sarkar. On the contrary, the parallelizable performance of IBTC is the same as that of Sarkar's and it doesn't have the same key length expansion with l -DIM and Shoup's construction.

8 Conclusion

In this paper we have provided two parallel domain extenders, IBTC and l -DIM, for UOWHF. Each of them has an important theoretical meaning in the study of efficient domain extending method for UOWHF.

IBTC has the most efficient key length expansion among all the previously known *complete* l -ary ($l \geq 2$) tree based parallel constructions. But IBTC need slightly more key length expansion than Shoup's sequential construction. On the other hand, l -DIM has the same key length expansion as Shoup's. Furthermore, l -DIM and Shoup's construction are the minimum possible for any algorithms in a large class of "natural" domain extenders including all the previously proposed constructions. But l -DIM does not have the same parallelizability performance as complete l -ary ($l \geq 2$) tree based constructions.

This paper has concerned the efficient parallel construction. Of course, it would be very nice to have a parallel construction which has the optimal key length expansion and the same or more efficient parallelizability than complete tree based constructions simultaneously. But at this point, we do not have any such algorithm. Hence, in our opinion, we should separately consider both the key length expansion and the parallelizability with the same importance. And we would like to stress that the present work is important in regarding the former and the latter point of views, respectively.

We have also given a sufficient condition for valid domain extension for sequential extension and it is likely that the condition is necessary. So, that will characterize the valid domain extension for sequential construction.

It is likely that l -DIM has maximum parallelizability with optimal key length expansion. So one can try to prove whether this parallelizability is maximum among all the constructions with optimal key length expansion or not.

Acknowledgments

We are grateful to Palash Sarkar for valuable discussions and unconditional encouragement. We also thank a number of anonymous referees for their helpful comments.

References

1. M. Bellare and P. Rogaway. *Collision-resistant hashing: towards making UOWHFs practical*, Advances in Cryptology - Crypto'97, Lecture Notes in Computer Science, Vol. 1294, Springer-Verlag, pp. 470-484, 1997.
2. I. B. Damgard. *A design principle for hash functions*, Advances in Cryptology - Crypto'89, Lecture Notes in Computer Sciences, Vol. 435, Springer-Verlag, pp. 416-427, 1989.
3. R. Merkle. *One way hash functions and DES*, Advances in Cryptology - Crypto'89, Lecture Notes in Computer Sciences, Vol. 435, Springer-Verlag, pp. 428-446, 1989.

4. I. Mironov. *Hash functions: from Merkle-Damgard to Shoup*, Advances in Cryptology - Eurocrypt'01, Lecture Notes in Computer Science, Vol. 2045, Springer-Verlag, pp 166-181, 2001
5. M. Nandi. *A New Tree based Domain Extension of UOWHF*, Cryptology ePrint Archive, <http://eprint.iacr.org/2003/142>.
6. M. Nandi. *Study of Domain Extension of UOWHF and its Optimality*, Cryptology ePrint Archive, <http://eprint.iacr.org/2003/158>.
7. M. Naor and M. Yung. *Universal one-way hash functions and their cryptographic applications*, Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing, ACM Press, pp 33-43, 1989.
8. P. Sarkar. *Construction of UOWHF: Tree Hashing Revisited*, Cryptology ePrint Archive, <http://eprint.iacr.org/2002/058>.
9. P. Sarkar. *Domain Extenders for UOWHF: A Generic Lower Bound on Key Expansion and a Finite Binary Tree Algorithm*, Cryptology ePrint Archive, <http://eprint.iacr.org/2003/009>.
10. V. Shoup. *A composition theorem for universal one-way hash functions*. Advances in Cryptology - Eurocrypt'00, Lecture Notes in Computer Science, Vol. 1807, Springer-Verlag, pp 445-452, 2000.
11. D. Simon. *Finding collisions on a one-way street: can secure hash functions be based on general assumptions?*, Advances in Cryptology - Eurocrypt'98, Lecture Notes in Computer Science, Vol. 1403, Springer-Verlag, pp 334-345, 1998.

Cryptanalysis of 3-Pass HAVAL[★]

Bart Van Rompay, Alex Biryukov, Bart Preneel^{★★}, and Joos Vandewalle

Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{bart.vanrompay,alex.biryukov,bart.preneel,joos.vandewalle}
@esat.kuleuven.ac.be

Abstract. HAVAL is a cryptographic hash function proposed in 1992 by Zheng, Pieprzyk and Seberry. Its has a structure that is quite similar to other well-known hash functions such as MD4 and MD5. The specification of HAVAL includes a security parameter: the number of passes (that is, the number of times that a particular word of the message is used in the computation) can be chosen equal to 3, 4 or 5. In this paper we describe a practical attack that finds collisions for the 3-pass version of HAVAL. This means that it is possible to generate pairs of messages hashing to the same value. The computational complexity of the attack corresponds to about 2^{29} computations of the compression function of 3-pass HAVAL; the required amount of memory is negligible.

1 Introduction

A cryptographic hash function is an algorithm that can be used to compress a message of arbitrary length into a hash value of specified length (say n bits). Such functions are widely used in applications requiring the authentication of information. In order to be useful for such applications it is required that the hash function is *one-way*: this means that, for a given value of n bits, it should be infeasible to find any message which hashes to this value. Another important property for a hash function is *collision-resistance*: it should be infeasible to find any two messages that are mapped by the function to the same value. This last property is not required in all applications of hash functions; one important case where it is needed is when a hash function is used in conjunction with a digital signature scheme, in order to compress a message before it is being signed.

Unfortunately one cannot design efficient hash functions with provable security properties. While it is possible to base a hash function on a different cryptographic primitive such as a block cipher (which may have received a lot of cryptanalytic effort and thereby confidence in its security), in practice dedicated algorithms, designed specifically for the purpose of hashing, are often preferred. Especially the algorithms of the so-called MD-family of hash functions are very

[★] This work was supported by the Concerted Research Action (GOA) Mefisto-2000/06 of the Flemish Government.

^{★★} Dr. Bart Preneel is professor at the Katholieke Universiteit Leuven, Belgium.

popular, because of their efficiency in software implementations and because of the experience gained by cryptanalysis of some members of this family.

The first algorithms of the MD-family were MD4 [10] and MD5 [11], proposed by Rivest in 1990 and 1991 respectively. These functions generate a hash value of 128 bits. The HAVAL [12] algorithm was proposed by Zheng, Pieprzyk and Seberry in 1992. In contrast to MD4 and MD5, HAVAL allows the computation of hashes of variable length, more specifically 128, 160, 192, 224 or 256 bits. This should result in higher security levels as the complexity of a collision-finding attack is conjectured to be of the order of $2^{n/2}$ operations where n is the number of bits in the hash value (this corresponds to the complexity of a generic birthday attack). The specification of HAVAL allows for a trade-off between efficiency and security margin by means of a parameter, the *number of passes*, which can be chosen equal to 3, 4 or 5. Amongst the other hash functions which belong to the MD-family are RIPEMD-160 [5] and SHA-1 [8], both of which have an output length of 160 bits. In order to generate longer hash values one can also use the recently proposed hash functions SHA-256, SHA-384 and SHA-512 [8] (with output length of 256, 384 or 512 bits respectively).

In 1996 Dobbertin [3] showed that the MD4 hash function is not collision-resistant: there is a practical attack that finds pairs of messages hashing to the same value. Later he applied similar techniques to find collisions for MD5 [4], but this attack does not work for the correct initial value defined for the algorithm (or for any other pre-specified initial value). In the case of HAVAL, only reduced versions of the algorithm have been analysed so far: it has been shown that collisions can be found when the number of passes is reduced to two [7,9,6]. In this paper we show a cryptanalysis of HAVAL in the case where the number of passes is equal to 3 (that is the minimum allowed by the algorithm specification). Our analysis leads to a practical attack that finds collisions for 3-pass HAVAL, using the correct initial value as specified for the algorithm, with a time complexity that corresponds to about 2^{29} computations of the compression function (this attack works for all possible output lengths of the algorithm). The remainder of the paper is organised as follows: in Section 2 we give a general outline of the attack procedure. The details of the attack are then explained in Sections 3 and 4. In Section 5 we provide a concrete example of a collision generated with our attack and we conclude in Section 6.

2 Outline of the Attack

The hash function HAVAL is defined as a simple iteration of a compression function and can be described as follows:

$$\mathcal{H}_0 = \mathcal{IV} \text{ , } \mathcal{H}_j = \text{compress}(\mathcal{H}_{j-1}, \mathcal{M}_j) \text{ (} 1 \leq j \leq t \text{) , } \text{hash}(\mathcal{M}) = \mathcal{H}_t \text{ .}$$

Here \mathcal{M} denotes the message which is divided into t blocks \mathcal{M}_j of 1024 bits each. \mathcal{IV} is an initial value of 256 bits, and \mathcal{H}_j represent chaining variables with a length of 256 bits. Each application of the compression function transforms the chaining variable into a new value under control of the current message

block \mathcal{M}_j , and the final value for this chaining variable serves as 256-bit hash value of the message \mathcal{M} . This construction implies that the problem of finding a collision for HAVAL can be reduced to the problem of finding a collision for its compression function. Note that an optional output transformation is defined for the computation of shorter hash values but this has no impact on our attack: we obtain a collision before the output transformation, therefore the attack works regardless of the length of the hash output.

In the following we will focus our attention on the compression function of HAVAL. This function uses only simple operations on 32-bit words. The 256-bit input is loaded into eight registers (A, B, C, D, E, F, G, H) and the 1024-bit message block is divided into 32 words $\{X_0, X_1, \dots, X_{31}\}$. Each step of the compression function updates the value of one of the registers, depending on a non-linear function of the other seven registers and also on one word of the message. For example the first step of the compression function updates the value of the A register in the following manner:

$$A \leftarrow A \gg^{11} + (f(B, C, D, E, F, G, H)) \gg^7 + X_0,$$

where f is a non-linear function; $(\cdot) \gg^s$ denotes rotation (circular shift) over s bit positions to the right, and $+$ denotes addition modulo 2^{32} . After 32 steps all words X_i have been used, and this constitutes the first pass of the HAVAL compression function. The 3-pass version has two more passes which again use all words X_i of the message exactly once (32 steps per pass) but the order in which they are applied is permuted. Also, each pass uses a different non-linear function in the step operations. We refer to the Appendix for a more detailed description of the compression function. We denote the values contained in the registers at the start of the compression function by (A_0, \dots, H_0) . Each pass of the compression function computes four new values for each register (4 values \times 8 registers = 32 steps). Hence, three passes compute 12 new values for the registers; these values are denoted (A_i, \dots, H_i) with $1 \leq i \leq 12$. Note that all steps of the compression function can be inverted, however there is a final feed-forward operation to make the function uninvertible. This operation computes the functions output as $(A_0 + A_{12}, \dots, H_0 + H_{12})$.

The goal of our attack is to find two distinct message blocks $\{X_i\}$ and $\{X'_i\}$ ($0 \leq i \leq 31$) which are mapped by the compression function to the same output value, where the computation for the two message block starts from the same 256-bit initial value (A_0, \dots, H_0) . We find such a collision for two message blocks with a small difference in only one of the words, more specifically:

$$\begin{aligned} X'_{28} &= X_{28} + 1, \\ X'_i &= X_i \quad (i \neq 28). \end{aligned}$$

During the execution of the compression function some intermediate values for the registers will be different for the message blocks $\{X_i\}$ and $\{X'_i\}$. We define the difference after step j as

$$\Delta_j = (A - A', B - B', C - C', D - D', E - E', F - F', G - G', H - H'),$$

where (A, \dots, H) are the contents of the registers at this point for message block $\{X_i\}$, and similarly (A', \dots, H') for $\{X'_i\}$. Note that this difference is defined with respect to the modular addition operation.

From the description of the compression function in the Appendix, it can be seen that the word X_{28} , respectively X'_{28} (which contains the only difference between the two message blocks) is applied three times, once in each of the three passes of the function. This is the case in steps 29, 38 and 69. Before step 29 all contents of the registers are equal for the two messages; a collision will be obtained if the contents of all registers are equal again after execution of step 69 (hereafter all message words that are used are the same for both messages so no new differences will occur in any computed register value). In order to give our attack a chance of success we need to control the differences in registers between step 29 and step 69 very carefully. The attack can be divided into two phases which we describe below and in more detail in the next sections.

Phase I: Inner Almost-Collision

The first phase of the attack concentrates on the first two passes of the compression function, more specifically the part between steps 29 and 38. The first use of the word X_{28} , respectively X'_{28} , is in step 29 (in pass 1 of the compression function) where a new value is computed for the E register. This means that the first computed register value which is not equal for the two messages, is the value E_4 , respectively E'_4 . At this point we have the following correspondence between the registers for the two messages:

$$\begin{array}{cccc} A_4 = A'_4 & B_4 = B'_4 & C_4 = C'_4 & D_4 = D'_4 \\ E_4 = E'_4 + (X_{28} - X'_{28}) & F_3 = F'_3 & G_3 = G'_3 & H_3 = H'_3 \end{array}$$

So the difference after step 29 is:

$$\Delta_{29} = (0, 0, 0, 0, X_{28} - X'_{28}, 0, 0, 0) = (0, 0, 0, 0, -1, 0, 0, 0).$$

The next use of X_{28} , respectively X'_{28} , occurs in step 38 (in pass 2 of the compression function) where a new value is computed for the F register. In this phase of the attack we fix some words X_i of the messages in such a way that we have the following correspondence between register values at this point:

$$\begin{array}{cccc} A_5 = A'_5 & B_5 = B'_5 & C_5 = C'_5 & D_5 = D'_5 \\ E_5 = E'_5 + 1 \ll^{12} & F_5 = F'_5 & G_4 = G'_4 & H_4 = H'_4 \end{array}$$

Here $(\cdot) \ll^s$ denotes rotation over s bit positions to the left. So we want only a small difference in register E after the execution of step 38. That is,

$$\Delta_{38} = (0, 0, 0, 0, 1 \ll^{12}, 0, 0, 0).$$

Such a set of differences $(\Delta_{29}, \Delta_{38})$ is called an *inner almost-collision*.

Phase II: Differential Analysis and Matching the Initial Value

The second phase of the attack concentrates on the last two passes of the compression function, more specifically the part between steps 38 and 69. As seen above we have only a small difference in the E register after step 38. We are now ready to perform a differential cryptanalysis on the following steps. The last occasion where the word X_{28} , respectively X'_{28} , is used is in step 69 (in pass 3 of the compression function). For $39 \leq j \leq 68$ we require that $\Delta_j = (0, 0, 0, 0, E - E', 0, 0, 0)$. That is, we require that the difference in the E register after step 38 does not spread to any of the other registers. Furthermore, the difference in the E register after step 38 has been chosen in such a way that the use of X_{28} , respectively X'_{28} , in step 69 compensates the difference in the E register at that point. That means $\Delta_{69} = (0, 0, 0, 0, 0, 0, 0, 0)$. This will also result in a collision in the output of the compression function.

In the previous phase of the attack we only needed to fix a few of the words X_i in the messages. Therefore, we can randomly choose the remaining words in this phase and see if the differential attack works. We found that the success probability of our differential attack is around 2^{-29} , so a collision can be found by randomly guessing the remaining words X_i and computing the difference after step 69 (which should be zero for all registers). This will succeed after, on average, 2^{29} trials.

There is one more complication to our attack: when all values of words X_i are determined we can calculate backwards in pass 1 of the compression function by inverting steps 29 down to 1. The values of (A_0, \dots, H_0) which we calculate in this way have to be equal to the initial values defined in the algorithm specification. This can be realised by randomly choosing only a subset of words X_i in this phase of the attack and calculating the values of some other words which can still be freely chosen so that the correct initial values are obtained.

3 Finding an Inner Almost-Collision

As noted in the previous section we first analyse the part of the compression function between step 29 and step 38. We require that $\Delta_{29} = (0, 0, 0, 0, -1, 0, 0, 0)$ and that $\Delta_{38} = (0, 0, 0, 0, 1^{\ll 12}, 0, 0, 0)$.

Table 1 below shows the difference propagation used in our attack. In step 29 a difference in the E register is introduced: $E_4 - E'_4 = X_{28} - X'_{28} = -1$. We let this difference spread to the F register in step 30, more specifically $F_4 - F'_4 = 1$. From step 31 up to 36 we require that the differences in the E and F registers do not spread to any of the other registers: $G_4 - G'_4 = H_4 - H'_4 = A_5 - A'_5 = B_5 - B'_5 = C_5 - C'_5 = D_5 - D'_5 = 0$. Then, in step 37 we need an interaction of the differences in the E and F registers, in such a way that the right difference $E_5 - E'_5 = 1^{\ll 12}$ is obtained. Finally, the difference in the F register has to disappear in step 38 where the word X_{28} , respectively X'_{28} , is used again: $F_5 - F'_5 = 0$.

For each step in turn, we now look at the difference which is obtained after computing the new register value for $\{X_i\}$ and $\{X'_i\}$. To simplify the analysis we first make the following specific choices:

$$E_4 = -1, E'_4 = 0, F_4 = 0, F'_4 = -1.$$

Note that these choices agree with the differences $E_4 - E'_4 = -1$ and $F_4 - F'_4 = 1$. The values 0 and -1 (modulo 2^{32}) correspond to 32-bit quantities where all the bits are set equal to 0 or 1 respectively.

Table 1. Overview of the difference propagation through the registers. The shown difference values are the values *after* the corresponding step has been executed. We also list the message word applied in each step. Note that $\Delta A = A - A'$, $\Delta B = B - B'$, etc. Entries in bold face show which register has been updated in a particular step.

Step	ΔA	ΔB	ΔC	ΔD	ΔE	ΔF	ΔG	ΔH	word
29	0	0	0	0	-1	0	0	0	$X_{28}(+1)$
30	0	0	0	0	-1	1	0	0	X_{29}
31	0	0	0	0	-1	1	0	0	X_{30}
32	0	0	0	0	-1	1	0	0	X_{31}
33	0	0	0	0	-1	1	0	0	X_5
34	0	0	0	0	-1	1	0	0	X_{14}
35	0	0	0	0	-1	1	0	0	X_{26}
36	0	0	0	0	-1	1	0	0	X_{18}
37	0	0	0	0	1 \ll^{12}	1	0	0	X_{11}
38	0	0	0	0	1 \ll^{12}	0	0	0	$X_{28}(+1)$

Step 29 In this step we have a difference in the applied message word X_{28} , respectively X'_{28} . From the definition of the step operation (see the Appendix) and using $E'_3 = E_3, F'_3 = F_3, G'_3 = G_3, H'_3 = H_3, A'_4 = A_4, B'_4 = B_4, C'_4 = C_4, D'_4 = D_4$ it follows that

$$E_4 - E'_4 = X_{28} - X'_{28} = -1.$$

Step 30 From the definition of the step operation it follows that

$$F_4 - F'_4 = (f(G_3, H_3, A_4, B_4, C_4, D_4, E_4)) \gg^7 - (f(G_3, H_3, A_4, B_4, C_4, D_4, E'_4)) \gg^7.$$

If we now use the definition of the non-linear function f (see the Appendix) and insert the values of E_4, E'_4, F_4, F'_4 we can rewrite this as

$$1 = (G_3 \oplus B_4 C_4 \oplus H_3 D_4 \oplus A_4 C_4 \oplus A_4) \gg^7 - (B_4 C_4 \oplus H_3 D_4 \oplus A_4 C_4 \oplus A_4) \gg^7. \quad (1)$$

Step 31 We require that $G_4 - G'_4 = 0$. That means,

$$(f(H_3, A_4, B_4, C_4, D_4, E_4, F_4)) \gg^7 - (f(H_3, A_4, B_4, C_4, D_4, E'_4, F'_4)) \gg^7 = 0.$$

Using the definition of f and inserting the values of E_4, E'_4, F_4, F'_4 we get

$$(A_4 \oplus C_4 D_4 \oplus B_4 D_4 \oplus B_4) \gg^7 = (H_3 \oplus C_4 D_4 \oplus B_4 D_4 \oplus B_4) \gg^7.$$

This equation is satisfied when

$$A_4 = H_3. \quad (2)$$

Step 32 We require that $H_4 - H'_4 = 0$. That means,

$$(f(A_4, B_4, C_4, D_4, E_4, F_4, G_4)) \gg^7 - (f(A_4, B_4, C_4, D_4, E'_4, F'_4, G_4)) \gg^7 = 0.$$

In the same manner as above we can derive the following equation:

$$D_4 \oplus C_4 = B_4. \quad (3)$$

Step 33 We require that $A_5 - A'_5 = 0$. Note that this is the first step of the second pass of the compression function so the non-linear function g is used (see the Appendix for the definition of the function g):

$$(g(B_4, C_4, D_4, E_4, F_4, G_4, H_4)) \gg^7 - (g(B_4, C_4, D_4, E'_4, F'_4, G_4, H_4)) \gg^7 = 0.$$

We obtain the equation

$$C_4 H_4 \oplus C_4 = C_4 G_4 \oplus H_4. \quad (4)$$

Step 34 We require that $B_5 - B'_5 = 0$. That means

$$(g(C_4, D_4, E_4, F_4, G_4, H_4, A_5)) \gg^7 - (g(C_4, D_4, E'_4, F'_4, G_4, H_4, A_5)) \gg^7 = 0,$$

which is satisfied when

$$D_4 A_5 \oplus H_4 = 0. \quad (5)$$

Step 35 We require that $C_5 - C'_5 = 0$. That means

$$(g(D_4, E_4, F_4, G_4, H_4, A_5, B_5)) \gg^7 - (g(D_4, E'_4, F'_4, G_4, H_4, A_5, B_5)) \gg^7 = 0,$$

which is satisfied when

$$G_4 B_5 \oplus H_4 A_5 \oplus G_4 \oplus D_4 = 0. \quad (6)$$

Step 36 We require that $D_5 - D'_5 = 0$. That means

$$(g(E_4, F_4, G_4, H_4, A_5, B_5, C_5)) \gg^7 - (g(E'_4, F'_4, G_4, H_4, A_5, B_5, C_5)) \gg^7 = 0,$$

which is satisfied when

$$H_4 C_5 \oplus A_5 B_5 \oplus H_4 \oplus G_4 = -1. \quad (7)$$

Step 37 In this step we need to obtain the right difference $E_5 - E'_5 = 1 \ll^{12}$. From the definition of the step operation it follows that

$$E_5 - E'_5 = E_4 \gg^{11} - E'_4 \gg^{11} + (g(F_4, G_4, H_4, A_5, B_5, C_5, D_5)) \gg^7 - (g(F'_4, G_4, H_4, A_5, B_5, C_5, D_5)) \gg^7.$$

Using the definition of g and inserting the values of E_4, E'_4, F_4, F'_4 we get

$$\begin{aligned} 1 \ll^{12} = & -1 + (G_4 A_5 D_5 \oplus G_4 B_5 C_5 \oplus G_4 A_5 \oplus A_5 C_5 \oplus G_4 H_4 \oplus B_5 D_5 \oplus \\ & B_5 C_5) \gg^7 - (G_4 A_5 D_5 \oplus G_4 B_5 C_5 \oplus G_4 A_5 \oplus A_5 C_5 \oplus G_4 H_4 \oplus B_5 D_5 \oplus \\ & B_5 C_5 \oplus G_4 \oplus -1) \gg^7. \end{aligned} \quad (8)$$

Step 38 Finally, in this step we require that the difference in the F register disappears: $F_5 - F'_5 = 0$. From the definition of the step operation we see that

$$F_5 - F'_5 = F_4^{\gg 11} - F_4'^{\gg 11} + X_{28} - X'_{28} + (g(G_4, H_4, A_5, B_5, C_5, D_5, E_5))^{\gg 7} - (g(G_4, H_4, A_5, B_5, C_5, D_5, E'_5))^{\gg 7}.$$

Because $F_4^{\gg 11} - F_4'^{\gg 11} = 1$ and $X_{28} - X'_{28} = -1$ the requirement $F_5 - F'_5 = 0$ leads to the equation

$$(g(G_4, H_4, A_5, B_5, C_5, D_5, E_5))^{\gg 7} - (g(G_4, H_4, A_5, B_5, C_5, D_5, E'_5))^{\gg 7} = 0,$$

which is satisfied when

$$B_5 H_4 \oplus C_5 = 0. \quad (9)$$

Solution for the System of Equations

The equations (1) to (9) which we derived above need to be satisfied in order to obtain an inner almost-collision. Therefore, we need a solution for an underdetermined system of 9 equations in 12 variables. It can be seen that the following set of register values constitutes such a solution:

$$\begin{array}{llllll} G_3 = 1^{\ll 7} & H_3 = 0 & A_4 = 0 & B_4 = 0 & C_4 = 0 & D_4 = 0 \\ G_4 = 0 & H_4 = 0 & A_5 = -1 & B_5 = -1 & C_5 = 0 & D_5 = 1^{\ll 18} \end{array}$$

Note that $G_3 = 1^{\ll 7}$ is a solution to $G_3^{\gg 7} = 1$, and $D_5 = 1^{\ll 18}$ is a solution to $-1 + D_5^{\gg 7} - (D_5 \oplus -1)^{\gg 7} = 1^{\ll 12}$. These two equations are derived from (1) and (8) respectively by inserting the values given for the other variables.

As previously seen we also have $E_4 = -1$ and $F_4 = 0$. Fixing these 14 register values, in order to generate an inner almost-collision, also determines the values of some words of the message block $\{X_i\}$. For example,

$$X_{30} = G_4 - G_3^{\gg 11} - (f(H_3, A_4, B_4, C_4, D_4, E_4, F_4))^{\gg 7}.$$

This follows from the definition of the step operation. In the same way, the message words X_{31} , X_5 , X_{14} , X_{26} , and X_{18} are determined. The values for these message words are as follows (in hexadecimal notation):

$$\begin{aligned} X_{30} &= \text{f0000000}_x \\ X_{31} &= \text{00000000}_x \\ X_5 &= \text{bad7de19}_x \\ X_{14} &= \text{c72fec88}_x \\ X_{26} &= \text{41ab9931}_x \\ X_{18} &= \text{cb1af394}_x \end{aligned}$$

Note that we get the same values $X'_i = X_i$ when we use the alternative register values $G'_3, H'_3, A'_4, B'_4, C'_4, D'_4, E'_4, F'_4, G'_4, H'_4, A'_5, B'_5, C'_5, D'_5$ in the computations (only E'_4 and F'_4 are different). Six words of the message blocks $\{X_i\}$ and $\{X'_i\}$ are now determined. We still have a free choice for the remaining 26 words of these message blocks in phase II of the attack, as described in Section 4.

Other Solutions for the System of Equations

As an alternative for the solution given above, different solutions for the system of equations (1) to (9) can be found. In general, for an arbitrary choice of two 32-bit values Q_1 and Q_2 , the following set of register values is a solution for the system of equations (and leads to an inner almost-collision):

$$\begin{array}{ll}
 G_3 = (1 + Q_1^{\gg 7})^{\ll 7} \oplus Q_1 & G_4 = (Q_2^{\gg 7} - 1^{\ll 12} - 1)^{\ll 7} \oplus Q_2 \oplus -1 \\
 H_3 = Q_1 & H_4 = 0 \\
 A_4 = Q_1 & A_5 = (Q_2^{\gg 7} - 1^{\ll 12} - 1)^{\ll 7} \oplus Q_2 \\
 B_4 = 0 & B_5 = -1 \\
 C_4 = 0 & C_5 = 0 \\
 D_4 = 0 & D_5 = Q_2
 \end{array}$$

Note that for $Q_1 = 0$ and $Q_2 = 1^{\ll 18}$ this reduces to the solution given earlier. For any choice of Q_1 and Q_2 a specific set of register values is obtained, and hence also a specific set of message words X_{30} , X_{31} , X_5 , X_{14} , X_{26} , and X_{18} . However, in those cases where bit 12 of Q_2 is equal to 1 (starting the count from the least significant bit position), the differential attack of Section 4 does not work. Solutions with bit 12 of Q_2 equal to 0 (leading to a successful differential attack), are called *admissible* inner almost-collisions. 2^{63} different admissible inner almost-collisions can be generated, but only one of them is needed for the next phase of the attack.

4 Differential Attack

In the second phase of the attack we perform a differential cryptanalysis (the technique of differential analysis was first applied to hash functions in [1]). We consider the part of the compression function between step 38 and step 69. We have an input difference $\Delta_{38} = (0, 0, 0, 0, 1^{\ll 12}, 0, 0, 0)$ (from the first phase of the attack) and require that $\Delta_{69} = (0, 0, 0, 0, 0, 0, 0, 0)$. Table 2 below shows the difference propagation for this phase of the attack. For the E register we have the following differences: $E_5 - E'_5 = 1^{\ll 12}$, $E_6 - E'_6 = 1^{\ll 1}$, $E_7 - E'_7 = 1^{\ll 22}$, $E_8 - E'_8 = 1^{\ll 11}$, $E_9 - E'_9 = 0$. For the other registers all differences must be zero.

There are two different cases for the computation of the probability of a difference propagation through a step. The content of the E register is updated in steps 45, 53, 61 and 69. In step 45 for example we compute

$$\begin{aligned}
 E_6 &= E_5^{\gg 11} + (g(F_5, G_5, H_5, A_6, B_6, C_6, D_6))^{\gg 7} + X_1 + K_{12}, \\
 E'_6 &= E'_5{}^{\gg 11} + (g(F_5, G_5, H_5, A_6, B_6, C_6, D_6))^{\gg 7} + X_1 + K_{12}.
 \end{aligned}$$

Hence, we see that the difference

$$E_6 - E'_6 = E_5^{\gg 11} - E'_5{}^{\gg 11},$$

and we require $E_5 - E'_5 = 1^{\ll 12}$ and $E_6 - E'_6 = 1^{\ll 1}$ (the difference gets rotated by 11 bit positions to the right). This happens with a probability which is close

Table 2. Overview of the difference propagation through the registers. The shown difference values are the values *after* the corresponding step has been executed. We also list the message word applied in each step. Note that $\Delta A = A - A'$, $\Delta B = B - B'$, etc. Entries in bold face show which register has been updated in a particular step.

Step	ΔA	ΔB	ΔC	ΔD	ΔE	ΔF	ΔG	ΔH	word
38	0	0	0	0	$1 \ll^{12}$	0	0	0	$X_{28(+1)}$
39	0	0	0	0	$1 \ll^{12}$	0	0	0	X_7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
44	0	0	0	0	$1 \ll^{12}$	0	0	0	X_{22}
45	0	0	0	0	$1 \ll^1$	0	0	0	X_1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
52	0	0	0	0	$1 \ll^1$	0	0	0	X_9
53	0	0	0	0	$1 \ll^{22}$	0	0	0	X_{17}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
60	0	0	0	0	$1 \ll^{22}$	0	0	0	X_{13}
61	0	0	0	0	$1 \ll^{11}$	0	0	0	X_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
68	0	0	0	0	$1 \ll^{11}$	0	0	0	X_{20}
69	0	0	0	0	0	0	0	0	$X_{28(+1)}$

to 1. In the other steps we require that the difference in the E register does not spread to a different register. In step 46 for example we compute

$$F_6 = F_5^{\gg 11} + (g(G_5, H_5, A_6, B_6, C_6, D_6, E_6))^{\gg 7} + X_1 + K_{12},$$

$$F'_6 = F_5^{\gg 11} + (g(G_5, H_5, A_6, B_6, C_6, D_6, E'_6))^{\gg 7} + X_1 + K_{12}.$$

Here the difference

$$F_6 - F'_6 = (g(G_5, H_5, A_6, B_6, C_6, D_6, E_6))^{\gg 7} - (g(G_5, H_5, A_6, B_6, C_6, D_6, E'_6))^{\gg 7},$$

and we require that $F_6 - F'_6 = 0$ which is equivalent to

$$g(G_5, H_5, A_6, B_6, C_6, D_6, E_6) = g(G_5, H_5, A_6, B_6, C_6, D_6, E'_6).$$

Using the definition of g we can derive the following condition:

$$E_6 B_6 H_5 \oplus E_6 C_6 = E'_6 B_6 H_5 \oplus E'_6 C_6,$$

which is satisfied when $B_6 H_5 \oplus C_6 = 0$ at those bit positions where E_6 is different from E'_6 . Because $E_6 = E'_6 + 1 \ll^1$ this happens with a probability of about $1/3$ ($\frac{1}{2^2} + \frac{1}{4^2} + \frac{1}{8^2} + \dots \approx \frac{1}{3}$).

By combining the probabilities for all steps we can estimate the global probability for the propagation from step 38 up to step 69 as $p_{69}^{38} \approx (1/3)^{27} \approx 2^{-42.8}$.

The real probability is much lower however. This is partly because of the contents of the registers at the start of the differential attack¹. Furthermore, the probabilities for consecutive steps strongly depend on each other (because every step changes the value of only 1 out of 8 registers). If we consider a sequence of 8 steps, experiments show that the probability is about 2^{-9} which is better than $(1/3)^7 \approx 2^{-11.1}$. For the complete propagation from step 38 to step 69 we found the estimation

$$p_{69}^{38} \approx 2^{-29}.$$

The differential attack can be performed as follows. In the previous section we saw that $X_{30}, X_{31}, X_5, X_{14}, X_{26}$, and X_{18} are determined in order to get the right input difference Δ_{38} . We can now randomly choose the remaining 26 words and calculate forwards to step 69, starting from the known register values $E_4, F_4, G_4, H_4, A_5, B_5, C_5, D_5$ (or E'_4, F'_4 for the second message block). If the difference after step 69 is equal to 0 for all registers then we have a collision and this happens on average after 2^{29} trials. There is one however one more complication which we describe below.

Matching the Initial Value

When all message words X_i are determined we can also compute backwards in pass 1 of the compression function, starting from the known register values $G_3, H_3, A_4, B_4, C_4, D_4, E_4, F_4$. This is done by inverting the step operations. For example, inverting step 30 gives us

$$F_3 = (F_4 - (f(G_3, H_3, A_4, B_4, C_4, D_4, E_4)))^{\gg 7} - X_{29})^{\ll 11}.$$

In that way we finally obtain the register values (A_0, \dots, H_0) . However these values should be equal to the initial values specified for the algorithm (see the Appendix). This can be solved as described below. First note that there is one sequence of 8 message words, which are applied in consecutive steps in pass 1 of the compression function, and none of which have been determined in phase I of the attack (for obtaining an inner almost-collision). This sequence of message words is the sequence of X_6, X_7, \dots, X_{13} (which is used in steps 7 to 14) and it will be used to match the correct initial values.

In our differential attack we randomly choose values for 18 message words (as before but excluding the 8 words needed to match the initial values). We also know the fixed values for the words $X_{30}, X_{31}, X_5, X_{14}, X_{26}, X_{18}$ (determined by phase I of the attack). Now we compute backwards in pass 1 of the compression function down to the (inverted) step 15 where X_{14} is applied. In this manner we derive the register values $(G_1, H_1, A_2, B_2, C_2, D_2, E_2, F_2)$. Next we compute forwards starting from the correct initial values and up to step 6 where X_5 is applied. This gives us the register values $(G_0, H_0, A_1, B_1, C_1, D_1, E_1, F_1)$ and

¹ Related to this, the reason that not all inner almost-collisions lead to a successful differential attack is that in some cases the contents of the registers are not suitable at the start of the differential attack.

now we can compute the required values for the message words X_6, X_7, \dots, X_{13} . For example,

$$X_6 = G_1 - G_0^{\gg 11} - (f(H_0, A_1, B_1, C_1, D_1, E_1, F_1))^{\gg 7}.$$

After we have matched the specified initial values for all registers (and thereby determined the values for all 32 message words X_i) we check the differential attack between steps 39 and 69 as before and repeat the procedure until a collision has been found. On average we succeed after 2^{29} trials, where a trial can be abandoned as soon as the difference propagation in a register is not correct. Note that the attack works equally well for the initial value specified for the algorithm or for any other initial value. A program that implements the attack runs on average in less than one hour on an Athlon 600MHz processor. Finally note that the number of collisions which can be generated, at least in theory, with this differential attack is equal to 2^{547} , since we can freely choose 18 words (that is a maximum of 2^{576} trials), and the success probability is about 2^{-29} . Because there are 2^{63} different admissible inner almost-collisions to start from, the total number of collisions which can be generated by our attack is equal to $2^{547+63} = 2^{610}$.

5 Example Collision for 3-Pass HAVAL

We give an example of two message blocks that are hashed by the compression function of 3-pass HAVAL to the same output value. This example has been checked using the reference implementation of HAVAL available at [2]. For both messages the computation starts from the initial value specified for the algorithm (this initial value is also used in [2]):

$$\begin{array}{llll} A_0 = \text{ec4e6c89}_x & B_0 = \text{082efa98}_x & C_0 = \text{299f31d0}_x & D_0 = \text{a4093822}_x \\ E_0 = \text{03707344}_x & F_0 = \text{13198a2e}_x & G_0 = \text{85a308d3}_x & H_0 = \text{243f6a88}_x \end{array}$$

The first message block is:

$$\begin{array}{llll} X_0 = \text{94c0875e}_x & X_1 = \text{dd25f63e}_x & X_2 = \text{f5d09361}_x & X_3 = \text{b51db8b2}_x \\ X_4 = \text{b00c36e4}_x & X_5 = \text{bad7de19}_x & X_6 = \text{32a68bb5}_x & X_7 = \text{c5aff25d}_x \\ X_8 = \text{ad0dea24}_x & X_9 = \text{a7e1ee7c}_x & X_{10} = \text{617b92dd}_x & X_{11} = \text{f9da283d}_x \\ X_{12} = \text{b2844d83}_x & X_{13} = \text{b8d498eb}_x & X_{14} = \text{c72fec88}_x & X_{15} = \text{8f467c05}_x \\ X_{16} = \text{507ea2c1}_x & X_{17} = \text{c2d94121}_x & X_{18} = \text{cb1af394}_x & X_{19} = \text{036daf20}_x \\ X_{20} = \text{bba7fb8c}_x & X_{21} = \text{6daee6aa}_x & X_{22} = \text{04fc029f}_x & X_{23} = \text{d37c05f4}_x \\ X_{24} = \text{993aea13}_x & X_{25} = \text{3ccfab88}_x & X_{26} = \text{41ab9931}_x & X_{27} = \text{3c7cae0c}_x \\ X_{28} = \text{f704bafc}_x & X_{29} = \text{b60635de}_x & X_{30} = \text{f0000000}_x & X_{31} = \text{00000000}_x \end{array}$$

and the second message block is determined by

$$\begin{aligned} X'_i &= X_i \quad (0 \leq i \leq 31, i \neq 28), \\ X'_{28} &= X_{28} + 1. \end{aligned}$$

For these two message blocks, the compression function computes the following common output value (note that this computation includes the feed-forward operation at the end):

$A = 1f46758c_x$ $B = 7618c292_x$ $C = e5220b62_x$ $D = 77ea845b_x$
 $E = ef9fd8de_x$ $F = 41ec28af_x$ $G = 5205cb85_x$ $H = 260412c4_x$

The complete hash function includes an additional application of the compression function, starting from the output value given above. For both messages the same padding block is used as message input for this final application of the compression function, therefore a collision is obtained in the final hash result:

$A = 7d476278_x$ $B = f603a907_x$ $C = 6d985fef_x$ $D = 4b5e66b7_x$
 $E = b6541db5_x$ $F = 16ccd71d_x$ $G = e8f9cf7c_x$ $H = 141e38e2_x$

Note that the algorithm converts this set of words into a string of 32 bytes, starting with the least significant byte of H and ending with the most significant byte of A (see the Appendix).

6 Conclusions

We have shown a practical attack for generating collisions in 3-pass HAVAL and believe that this version of HAVAL should no longer be used in applications where a collision-resistant hash function is required. The strategy for our attack is quite similar to the strategy that was used for the cryptanalysis of MD4 in [3]. Surprisingly, our result shows that the use of highly non-linear functions, which is the main focus of the design of HAVAL, does not result in a hash function which is significantly stronger compared to MD4 (note that MD4's compression function also has 3 passes but only 16 steps in each pass). We believe that it may be possible to extend our techniques in order to generate predictable output differences in the 4-pass version of HAVAL but further research is needed to examine this.

References

1. E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
2. Calyptix Security, "HAVAL source code (reference implementation)", available at <http://www.calyptix.com/downloads.html>
3. H. Dobbertin, "Cryptanalysis of MD4," *Fast Software Encryption '96, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 53–69.
4. H. Dobbertin, "The status of MD5 after a recent attack," *Cryptobytes, vol. 2, no. 2*, Summer 1996, pp. 1–6.
5. H. Dobbertin, A. Bosselaers and B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD," *Fast Software Encryption '96, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 71–82.
6. Y.-S. Her, K. Sakurai and S.-H. Kim, "Attacks for finding collision in reduced versions of 3-pass and 4-pass HAVAL," *Proceedings International Conference on Computers, Communications and Systems (2003ICCCS)*, CE-15, pp. 75–78.
7. P. Kasselmann and W. Penzhorn, "Cryptanalysis of reduced version of HAVAL", *Electronics letters, vol. 36, no. 1*, January 2000, pp. 30–31.

8. National Institute of Standards and Technology, FIPS-180-2: *Secure Hash Standard (SHS)*, August 2002.
9. S. Park, S. H. Sung, S. Chee, J. Lim, “On the security of reduced versions of 3-pass HAVAL,” *Proceedings of ACISP 2002*, pp. 406–419.
10. R.L. Rivest, “The MD4 message-digest algorithm,” *Advances in Cryptology – Crypto’90, LNCS 537*, A. Menezes and S. Vanstone, Eds., Springer-Verlag, 1990, pp. 303–311.
11. R.L. Rivest, “The MD5 message-digest algorithm,” *Request for Comments (RFC) 1321*, Internet Activities Board, Internet Privacy Task Force, April 1992.
12. Y. Zheng, J. Pieprzyk and J. Seberry, “HAVAL – a one-way hashing algorithm with variable length of output,” *Advances in Cryptology – AusCrypt ’92, LNCS 718*, J. Seberry and Y. Zheng, Eds., Springer-Verlag, 1993, pp. 83–104.

Appendix

In this appendix we give a description of HAVAL and explain the notations that are used in this paper. Not all of the details are fully described: for a complete specification see [12]. HAVAL is defined as the iteration of a compression function which we specify below. Each application of this compression function uses eight words as initial value and 32 words of the message as input, and produces eight words of output which are then used as initial value for the next application of the compression function. All words have a length of 32 bits (4 bytes). The initial value to be used in the first application of the compression function is specified as follows (hexadecimal notation):

$$IV = \text{ec4e6c89}_x \text{ 082efa98}_x \text{ 299f31d0}_x \text{ a4093822}_x \\ \text{03707344}_x \text{ 13198a2e}_x \text{ 85a308d3}_x \text{ 243f6a88}_x .$$

Note that there is a padding rule that appends bytes to the message so that its length becomes a multiple of 128 bytes (32 words \times 4 bytes/word). The added bytes include a representation of the length of the original message. The little endian-convention is used to transform the message (sequence of bytes) into a sequence of words.

The compression function uses three non-linear functions, each of which takes seven words of input and produces one word of output:

$$\begin{aligned} f(Z_6, Z_5, Z_4, Z_3, Z_2, Z_1, Z_0) &= Z_2Z_3 \oplus Z_6Z_0 \oplus Z_5Z_1 \oplus Z_4Z_2 \oplus Z_4, \\ g(Z_6, Z_5, Z_4, Z_3, Z_2, Z_1, Z_0) &= Z_3Z_5Z_0 \oplus Z_5Z_1Z_2 \oplus Z_3Z_5 \oplus Z_3Z_1 \oplus \\ &\quad Z_5Z_4 \oplus Z_0Z_2 \oplus Z_1Z_2 \oplus Z_6Z_5 \oplus Z_6, \\ h(Z_6, Z_5, Z_4, Z_3, Z_2, Z_1, Z_0) &= Z_5Z_4Z_3 \oplus Z_5Z_2 \oplus Z_4Z_1 \oplus Z_3Z_6 \oplus Z_0Z_3 \oplus Z_0. \end{aligned}$$

Here Z_iZ_j denotes the Boolean AND function of Z_i and Z_j , and $Z_i \oplus Z_j$ denotes the Boolean exclusive-OR function of Z_i and Z_j . Note that the functions f , g and h operate at bit-level: they can be performed independently at each of the 32 bit positions in the words.

Let $ff(Z_7, Z_6, Z_5, Z_4, Z_3, Z_2, Z_1, Z_0, X)$, $gg(Z_7, Z_6, Z_5, Z_4, Z_3, Z_2, Z_1, Z_0, X)$ and $hh(Z_7, Z_6, Z_5, Z_4, Z_3, Z_2, Z_1, Z_0, X)$ be equivalent to

$$\begin{aligned} Z_7^{\gg 11} + (f(Z_6, Z_5, Z_4, Z_3, Z_2, Z_1, Z_0))^{\gg 7} + X, \\ Z_7^{\gg 11} + (g(Z_6, Z_5, Z_4, Z_3, Z_2, Z_1, Z_0))^{\gg 7} + X, \\ Z_7^{\gg 11} + (h(Z_6, Z_5, Z_4, Z_3, Z_2, Z_1, Z_0))^{\gg 7} + X, \end{aligned}$$

where $(\cdot)^{\gg s}$ denotes rotation (circular shift) over s bit positions to the right, and $+$ denotes addition modulo 2^{32} .

Suppose that the initial value $(A_0, B_0, C_0, D_0, E_0, F_0, G_0, H_0)$ is given. Then the compression function applies the following 96 steps (three passes of 32 steps each):

PASS 1

STEP

$$A_1 = ff(A_0, B_0, C_0, D_0, E_0, F_0, G_0, H_0, X_0) \quad (1)$$

$$B_1 = ff(B_0, C_0, D_0, E_0, F_0, G_0, H_0, A_1, X_1) \quad (2)$$

$$C_1 = ff(C_0, D_0, E_0, F_0, G_0, H_0, A_1, B_1, X_2) \quad (3)$$

$$D_1 = ff(D_0, E_0, F_0, G_0, H_0, A_1, B_1, C_1, X_3) \quad (4)$$

$$E_1 = ff(E_0, F_0, G_0, H_0, A_1, B_1, C_1, D_1, X_4) \quad (5)$$

$$F_1 = ff(F_0, G_0, H_0, A_1, B_1, C_1, D_1, E_1, X_5) \quad (6)$$

$$G_1 = ff(G_0, H_0, A_1, B_1, C_1, D_1, E_1, F_1, X_6) \quad (7)$$

$$H_1 = ff(H_0, A_1, B_1, C_1, D_1, E_1, F_1, G_1, X_7) \quad (8)$$

$$A_2 = ff(A_1, B_1, C_1, D_1, E_1, F_1, G_1, H_1, X_8) \quad (9)$$

$$B_2 = ff(B_1, C_1, D_1, E_1, F_1, G_1, H_1, A_2, X_9) \quad (10)$$

$$C_2 = ff(C_1, D_1, E_1, F_1, G_1, H_1, A_2, B_2, X_{10}) \quad (11)$$

$$D_2 = ff(D_1, E_1, F_1, G_1, H_1, A_2, B_2, C_2, X_{11}) \quad (12)$$

$$E_2 = ff(E_1, F_1, G_1, H_1, A_2, B_2, C_2, D_2, X_{12}) \quad (13)$$

$$F_2 = ff(F_1, G_1, H_1, A_2, B_2, C_2, D_2, E_2, X_{13}) \quad (14)$$

$$G_2 = ff(G_1, H_1, A_2, B_2, C_2, D_2, E_2, F_2, X_{14}) \quad (15)$$

$$H_2 = ff(H_1, A_2, B_2, C_2, D_2, E_2, F_2, G_2, X_{15}) \quad (16)$$

$$A_3 = ff(A_2, B_2, C_2, D_2, E_2, F_2, G_2, H_2, X_{16}) \quad (17)$$

$$B_3 = ff(B_2, C_2, D_2, E_2, F_2, G_2, H_2, A_3, X_{17}) \quad (18)$$

$$C_3 = ff(C_2, D_2, E_2, F_2, G_2, H_2, A_3, B_3, X_{18}) \quad (19)$$

$$D_3 = ff(D_2, E_2, F_2, G_2, H_2, A_3, B_3, C_3, X_{19}) \quad (20)$$

$$E_3 = ff(E_2, F_2, G_2, H_2, A_3, B_3, C_3, D_3, X_{20}) \quad (21)$$

$$F_3 = ff(F_2, G_2, H_2, A_3, B_3, C_3, D_3, E_3, X_{21}) \quad (22)$$

$$G_3 = ff(G_2, H_2, A_3, B_3, C_3, D_3, E_3, F_3, X_{22}) \quad (23)$$

$$H_3 = ff(H_2, A_3, B_3, C_3, D_3, E_3, F_3, G_3, X_{23}) \quad (24)$$

$$A_4 = ff(A_3, B_3, C_3, D_3, E_3, F_3, G_3, H_3, X_{24}) \quad (25)$$

$$B_4 = ff(B_3, C_3, D_3, E_3, F_3, G_3, H_3, A_4, X_{25}) \quad (26)$$

$$C_4 = ff(C_3, D_3, E_3, F_3, G_3, H_3, A_4, B_4, X_{26}) \quad (27)$$

$$D_4 = ff(D_3, E_3, F_3, G_3, H_3, A_4, B_4, C_4, X_{27}) \quad (28)$$

$$E_4 = ff(E_3, F_3, G_3, H_3, A_4, B_4, C_4, D_4, X_{28}) \quad (29)$$

$$F_4 = ff(F_3, G_3, H_3, A_4, B_4, C_4, D_4, E_4, X_{29}) \quad (30)$$

$$G_4 = ff(G_3, H_3, A_4, B_4, C_4, D_4, E_4, F_4, X_{30}) \quad (31)$$

$$H_4 = ff(H_3, A_4, B_4, C_4, D_4, E_4, F_4, G_4, X_{31}) \quad (32)$$

PASS 2

STEP

$$A_5 = gg(A_4, B_4, C_4, D_4, E_4, F_4, G_4, H_4, X_5 + K_0) \quad (33)$$

$$B_5 = gg(B_4, C_4, D_4, E_4, F_4, G_4, H_4, A_5, X_{14} + K_1) \quad (34)$$

$$C_5 = gg(C_4, D_4, E_4, F_4, G_4, H_4, A_5, B_5, X_{26} + K_2) \quad (35)$$

$$D_5 = gg(D_4, E_4, F_4, G_4, H_4, A_5, B_5, C_5, X_{18} + K_3) \quad (36)$$

$$E_5 = gg(E_4, F_4, G_4, H_4, A_5, B_5, C_5, D_5, X_{11} + K_4) \quad (37)$$

$$F_5 = gg(F_4, G_4, H_4, A_5, B_5, C_5, D_5, E_5, X_{28} + K_5) \quad (38)$$

$$G_5 = gg(G_4, H_4, A_5, B_5, C_5, D_5, E_5, F_5, X_7 + K_6) \quad (39)$$

$$H_5 = gg(H_4, A_5, B_5, C_5, D_5, E_5, F_5, G_5, X_{16} + K_7) \quad (40)$$

$$A_6 = gg(A_5, B_5, C_5, D_5, E_5, F_5, G_5, H_5, X_0 + K_8) \quad (41)$$

$$B_6 = gg(B_5, C_5, D_5, E_5, F_5, G_5, H_5, A_6, X_{23} + K_9) \quad (42)$$

$$C_6 = gg(C_5, D_5, E_5, F_5, G_5, H_5, A_6, B_6, X_{20} + K_{10}) \quad (43)$$

$$D_6 = gg(D_5, E_5, F_5, G_5, H_5, A_6, B_6, C_6, X_{22} + K_{11}) \quad (44)$$

$$E_6 = gg(E_5, F_5, G_5, H_5, A_6, B_6, C_6, D_6, X_1 + K_{12}) \quad (45)$$

$$F_6 = gg(F_5, G_5, H_5, A_6, B_6, C_6, D_6, E_6, X_{10} + K_{13}) \quad (46)$$

$$G_6 = gg(G_5, H_5, A_6, B_6, C_6, D_6, E_6, F_6, X_4 + K_{14}) \quad (47)$$

$$H_6 = gg(H_5, A_6, B_6, C_6, D_6, E_6, F_6, G_6, X_8 + K_{15}) \quad (48)$$

$$A_7 = gg(A_6, B_6, C_6, D_6, E_6, F_6, G_6, H_6, X_{30} + K_{16}) \quad (49)$$

$$B_7 = gg(B_6, C_6, D_6, E_6, F_6, G_6, H_6, A_7, X_3 + K_{17}) \quad (50)$$

$$C_7 = gg(C_6, D_6, E_6, F_6, G_6, H_6, A_7, B_7, X_{21} + K_{18}) \quad (51)$$

$$D_7 = gg(D_6, E_6, F_6, G_6, H_6, A_7, B_7, C_7, X_9 + K_{19}) \quad (52)$$

$$E_7 = gg(E_6, F_6, G_6, H_6, A_7, B_7, C_7, D_7, X_{17} + K_{20}) \quad (53)$$

$$F_7 = gg(F_6, G_6, H_6, A_7, B_7, C_7, D_7, E_7, X_{24} + K_{21}) \quad (54)$$

$$G_7 = gg(G_6, H_6, A_7, B_7, C_7, D_7, E_7, F_7, X_{29} + K_{22}) \quad (55)$$

$$H_7 = gg(H_6, A_7, B_7, C_7, D_7, E_7, F_7, G_7, X_6 + K_{23}) \quad (56)$$

$$A_8 = gg(A_7, B_7, C_7, D_7, E_7, F_7, G_7, H_7, X_{19} + K_{24}) \quad (57)$$

$$B_8 = gg(B_7, C_7, D_7, E_7, F_7, G_7, H_7, A_8, X_{12} + K_{25}) \quad (58)$$

$$C_8 = gg(C_7, D_7, E_7, F_7, G_7, H_7, A_8, B_8, X_{15} + K_{26}) \quad (59)$$

$$D_8 = gg(D_7, E_7, F_7, G_7, H_7, A_8, B_8, C_8, X_{13} + K_{27}) \quad (60)$$

$$E_8 = gg(E_7, F_7, G_7, H_7, A_8, B_8, C_8, D_8, X_2 + K_{28}) \quad (61)$$

$$F_8 = gg(F_7, G_7, H_7, A_8, B_8, C_8, D_8, E_8, X_{25} + K_{29}) \quad (62)$$

$$G_8 = gg(G_7, H_7, A_8, B_8, C_8, D_8, E_8, F_8, X_{31} + K_{30}) \quad (63)$$

$$H_8 = gg(H_7, A_8, B_8, C_8, D_8, E_8, F_8, G_8, X_{27} + K_{31}) \quad (64)$$

PASS 3

STEP

$$A_9 = hh(A_8, B_8, C_8, D_8, E_8, F_8, G_8, H_8, X_{19} + K_{32}) \quad (65)$$

$$B_9 = hh(B_8, C_8, D_8, E_8, F_8, G_8, H_8, A_9, X_9 + K_{33}) \quad (66)$$

$$C_9 = hh(C_8, D_8, E_8, F_8, G_8, H_8, A_9, B_9, X_4 + K_{34}) \quad (67)$$

$$D_9 = hh(D_8, E_8, F_8, G_8, H_8, A_9, B_9, C_9, X_{20} + K_{35}) \quad (68)$$

$$E_9 = hh(E_8, F_8, G_8, H_8, A_9, B_9, C_9, D_9, X_{28} + K_{36}) \quad (69)$$

$$F_9 = hh(F_8, G_8, H_8, A_9, B_9, C_9, D_9, E_9, X_{17} + K_{37}) \quad (70)$$

$$G_9 = hh(G_8, H_8, A_9, B_9, C_9, D_9, E_9, F_9, X_8 + K_{38}) \quad (71)$$

$$H_9 = hh(H_8, A_9, B_9, C_9, D_9, E_9, F_9, G_9, X_{22} + K_{39}) \quad (72)$$

$$A_{10} = hh(A_9, B_9, C_9, D_9, E_9, F_9, G_9, H_9, X_{29} + K_{40}) \quad (73)$$

$$B_{10} = hh(B_9, C_9, D_9, E_9, F_9, G_9, H_9, A_{10}, X_{14} + K_{41}) \quad (74)$$

$$C_{10} = hh(C_9, D_9, E_9, F_9, G_9, H_9, A_{10}, B_{10}, X_{25} + K_{42}) \quad (75)$$

$$D_{10} = hh(D_9, E_9, F_9, G_9, H_9, A_{10}, B_{10}, C_{10}, X_{12} + K_{43}) \quad (76)$$

$$E_{10} = hh(E_9, F_9, G_9, H_9, A_{10}, B_{10}, C_{10}, D_{10}, X_{24} + K_{44}) \quad (77)$$

$$F_{10} = hh(F_9, G_9, H_9, A_{10}, B_{10}, C_{10}, D_{10}, E_{10}, X_{30} + K_{45}) \quad (78)$$

$$G_{10} = hh(G_9, H_9, A_{10}, B_{10}, C_{10}, D_{10}, E_{10}, F_{10}, X_{16} + K_{46}) \quad (79)$$

$$H_{10} = hh(H_9, A_{10}, B_{10}, C_{10}, D_{10}, E_{10}, F_{10}, G_{10}, X_{26} + K_{47}) \quad (80)$$

$$A_{11} = hh(A_{10}, B_{10}, C_{10}, D_{10}, E_{10}, F_{10}, G_{10}, H_{10}, X_{31} + K_{48}) \quad (81)$$

$$B_{11} = hh(B_{10}, C_{10}, D_{10}, E_{10}, F_{10}, G_{10}, H_{10}, A_{11}, X_{15} + K_{49}) \quad (82)$$

$$C_{11} = hh(C_{10}, D_{10}, E_{10}, F_{10}, G_{10}, H_{10}, A_{11}, B_{11}, X_7 + K_{50}) \quad (83)$$

$$D_{11} = hh(D_{10}, E_{10}, F_{10}, G_{10}, H_{10}, A_{11}, B_{11}, C_{11}, X_3 + K_{51}) \quad (84)$$

$$E_{11} = hh(E_{10}, F_{10}, G_{10}, H_{10}, A_{11}, B_{11}, C_{11}, D_{11}, X_1 + K_{52}) \quad (85)$$

$$F_{11} = hh(F_{10}, G_{10}, H_{10}, A_{11}, B_{11}, C_{11}, D_{11}, E_{11}, X_0 + K_{53}) \quad (86)$$

$$G_{11} = hh(G_{10}, H_{10}, A_{11}, B_{11}, C_{11}, D_{11}, E_{11}, F_{11}, X_{18} + K_{54}) \quad (87)$$

$$H_{11} = hh(H_{10}, A_{11}, B_{11}, C_{11}, D_{11}, E_{11}, F_{11}, G_{11}, X_{27} + K_{55}) \quad (88)$$

$$A_{12} = hh(A_{11}, B_{11}, C_{11}, D_{11}, E_{11}, F_{11}, G_{11}, H_{11}, X_{13} + K_{56}) \quad (89)$$

$$B_{12} = hh(B_{11}, C_{11}, D_{11}, E_{11}, F_{11}, G_{11}, H_{11}, A_{12}, X_6 + K_{57}) \quad (90)$$

$$C_{12} = hh(C_{11}, D_{11}, E_{11}, F_{11}, G_{11}, H_{11}, A_{12}, B_{12}, X_{21} + K_{58}) \quad (91)$$

$$D_{12} = hh(D_{11}, E_{11}, F_{11}, G_{11}, H_{11}, A_{12}, B_{12}, C_{12}, X_{10} + K_{59}) \quad (92)$$

$$E_{12} = hh(E_{11}, F_{11}, G_{11}, H_{11}, A_{12}, B_{12}, C_{12}, D_{12}, X_{23} + K_{60}) \quad (93)$$

$$F_{12} = hh(F_{11}, G_{11}, H_{11}, A_{12}, B_{12}, C_{12}, D_{12}, E_{12}, X_{11} + K_{61}) \quad (94)$$

$$G_{12} = hh(G_{11}, H_{11}, A_{12}, B_{12}, C_{12}, D_{12}, E_{12}, F_{12}, X_5 + K_{62}) \quad (95)$$

$$H_{12} = hh(H_{11}, A_{12}, B_{12}, C_{12}, D_{12}, E_{12}, F_{12}, G_{12}, X_2 + K_{63}) \quad (96)$$

The values K_i used in the last two passes are 32-bit constants derived from the fractional part of π . Finally, the eight-word output of the compression function is computed with a feed-forward of the initial value:

$$\begin{aligned} A &= A_0 + A_{12} & B &= B_0 + B_{12} & C &= C_0 + C_{12} & D &= D_0 + D_{12} \\ E &= E_0 + E_{12} & F &= F_0 + F_{12} & G &= G_0 + G_{12} & H &= H_0 + H_{12} \end{aligned}$$

The obtained words (A, B, C, D, E, F, G, H) serve as initial value for the next application of the compression function. If this was the final use of the compression function (the last 32 words of the padded message have been processed), the concatenated 256-bit value $H\|G\|F\|E\|D\|C\|B\|A$ serves as hash value of the message, where the little endian-convention is used to transform the sequence of words into a sequence of bytes (the first byte is the least significant byte of H and the last byte is the most significant byte of A). There is an optional output transformation which allows to reduce the length of this hash value to 128, 160, 192 or 224 bits.

Efficient Group Signatures without Trapdoors^{*}

Giuseppe Ateniese and Breno de Medeiros

The Johns Hopkins University
Department of Computer Science
Baltimore, MD 21218, USA

ateniese@cs.jhu.edu, breno.demedeiros@acm.org

Abstract. Group signature schemes are fundamental cryptographic tools that enable unlinkably anonymous authentication, in the same fashion that digital signatures provide the basis for strong authentication protocols. In this paper we present the first group signature scheme with constant-size parameters that does not require any group member, including group managers, to know trapdoor secrets. This novel type of group signature scheme allows public parameters to be shared among organizations. Such sharing represents a highly desirable simplification over existing schemes, which require each organization to maintain a separate cryptographic domain.

Keywords: Group signatures, privacy and anonymity, cryptographic protocols.

1 Introduction

Group signatures allow group members to anonymously sign arbitrary messages on behalf of the group. In addition, signatures generated from the same signer are unlinkable, i.e., it is difficult to determine whether two or more signatures were generated by the same group member. In case of dispute, a group manager will be able to *open* a signature and incontestably show the identity of the signer. At the same time, no one (including the group manager) will be able to falsely accuse any other member of the group.

Group signatures were introduced by D. Chaum and E. van Heyst [16] in 1991. That was followed by several other works, but only relatively recent ones [3,10,11] have group public keys and group signatures with sizes that do not depend on the number of group members. (While in theory one always needs at least $\log n$ bits to uniquely identify n different users in any system, in practice $\log n$ is orders of magnitude smaller than the bit length of keys used in public key cryptography.) The scheme in [3] is the most efficient one and the only proven secure against an adaptive adversary. However, all the existing group signature schemes providing constant-size parameters require the group manager to know the factors of an RSA modulus. Sharing these factors among group managers of different organizations would compromise the security and/or the

^{*} This work was partly funded by NSF.

trust assumptions of the entire scheme. This paper provides the first, affirmative answer to the question of whether it is possible to design trapdoor-free group signature schemes with public parameters that do not increase linearly in size with the number of group members. We have an informal proof of security for the scheme (along the lines of the proof in [3]), and sketch some arguments that might lead to a formal proof in the sense of [5], in appendix §B.

1.1 Motivation

Our schemes are useful when several distinct groups or organizations must interact and exchange information about individuals while protecting their privacy. Credential transfer systems (CTS) [14,15,19,17,23,9] are examples of such environments that can be built via group signature schemes [9]. Real-world scenarios for the use of CTS include the health-care industry, electronic voting, and transportation systems. In such cases, the added manageability and improved optimization opportunities permitted by the use of a single cryptographic domain for all participating organizations may outweigh other efficiency considerations. A CTS allows users to interact anonymously with several organizations so that it is possible to prove possession of a credential from one organization to another. Different transactions cannot be linked to real identities or even pseudonyms. It is then impossible to create profiles of users even if the organizations collude and, at the same time, users cannot falsely claim to possess credentials. Optionally, a privacy officer is able to retrieve user identities in case of disputes or emergencies. Users can thus authenticate themselves with anonymous credentials, protecting their privacy while exercising their right to vote, obtaining health services or renting a GPS-tracked automobile. The efficiency of a single signature generation or verification is measured in the human time scale. Consequently, theoretical computational advantages become less important, and instead the administrative complexity and related costs are likely to be the overwhelming concern of implementers. In these situations, a scheme with shareable parameters has a definite advantage since it eliminates the need for specialized techniques such as the ones employed in [9].

Recently in [5], it has been shown that group signatures can be built based on the assumption that trapdoor functions exist. It would be interesting to show the same but based on the existence of one-way functions. Our scheme is the first to be functionally trapdoor-free as no group member, nor even the group manager, needs to know the trapdoor information. Even though we use an RSA ring and we rely on the strong RSA assumption for security, the operation of the scheme exploits only the one-wayness of the RSA function, not its trapdoor properties.

Organization of This Paper: The next section contains the definition of group signatures and the attending security requirements. In section §3 we give a high-level, intuitive description of our proposed scheme, and place it in the context of previous work. That section also introduces the cryptographic building blocks required for the scheme. The specific construction of our scheme takes all of section §4. A security analysis is provided in appendix §B.

2 Definition

In this section we present our characterization of group signature schemes. In general, a group signature scheme is defined by a family of procedures:

SETUP: A probabilistic algorithm that generates the group-specific parameters. The input to **SETUP** is the set of public parameters, which includes a security parameter, and its output are the group public key \mathcal{P} and associated secret key \mathcal{S} .

JOIN: A prospective member executes this protocol (interacting with the group manager) to join the group. The new member's output is a membership certificate and the corresponding secret.

SIGN: A probabilistic algorithm that outputs a group signature when given as input a message, the group public key, a membership certificate, and the associated membership secret.

VERIFY: A boolean-valued algorithm used to test the authenticity of signatures generated by **SIGN**.

OPEN: An algorithm that given as input a message, a group signature on it, and the group secret key, extracts the membership certificate used to issue the signature, and a non-interactive proof of the signature's authorship.

2.1 Properties Required

A group signature scheme must satisfy the following properties:

Correctness: A properly formed group signature must be accepted by the verification algorithm.

Unforgeability: Without possession of a membership certificate, and knowledge of associated secret, it is computationally infeasible to produce a signature that is accepted by the verification algorithm.

Anonymity/ Unlinkability: Given a group signature on a message, it is computationally infeasible to determine which member generated the signature. Moreover, given several group signatures on the same or different messages it is computationally infeasible to decide whether the signatures were issued by the same or by different group members.

Exculpability: A signature produced by a group member cannot be successfully attributed to another, and the group manager cannot generate signatures on behalf of other group members (**non-framing**).

Traceability: The group manager is "always" (with overwhelming probability) able to open a valid signature and determine which member signed it. Even if a coalition of group members collaborates to produce a signature on a message, possibly by combining their certificate secrets in some fashion, the group manager will succeed in attributing the signature to one of the colluding members (**coalition-resistance**) [3].

The requirements of unforgeability and coalition-resistance are equivalent to the requirements that group membership certificates be unforgeable under passive and active attacks, respectively, and only issuable by the group manager. In other words, a membership certificate should contain the equivalent of a digital signature by the group manager. Similarly, the requirements of traceability and exculpability imply that the group signature should hide a regular digital signature issued by the member.

These listed requirements are intuitive, but somewhat redundant: For instance, exculpability and traceability are clearly connected. In [5] the first formal model of group signature schemes was introduced, showing the relations between different requirements, and simplifying the task of proving the security of a group signature scheme. In that work, the authors claim that all security requirements of group signature schemes are derivable from two newly defined concepts: *full anonymity* and *full traceability*.

The new model introduces *two* independent group managers, one in charge of group membership management tasks, such as adding to or removing members from the group, and another responsible for opening group signatures – i.e., revealing the identity of the signer. The first manager provides *privacy* by enabling users to sign and authenticate themselves anonymously (or more properly, as arbitrary group members), while the second manager provides *accountability*, by tracing authorship of group signatures back to the issuer when required. Compromise of the first manager’s secret key permits one to enroll arbitrary signing keys in the group and issue signatures on behalf of these non-entities. However it does not allow one to trace authorship of signatures. Compromise of the second manager’s secret key allows one to trace authorship of signatures, but not to add new public keys to the group.

Definition 1. *Full anonymity (cf [5]): This is defined in terms of an adversarial game. The goal of the adversary is to defeat the anonymity by identifying the authorship of a group signature on a message. The game takes place in two stages. In the first (choose) stage, the adversary is given access to all members’ secret keys. It also has access to an OPEN oracle, which it can query to find the authorship of various group signatures. The output of the first stage is two member identities i_0 and i_1 , a message m and some state information S . These are given as input to the second (guess) stage, in which the adversary is also given a group signature σ on m , which is known to have been issued by either i_0 or i_1 with equal probability. The adversary can continue to query the OPEN oracle on signatures other than σ . The output of this stage is a guess i_b for the identity of the signer. The adversary is said to win this game if it can guess the correct signer with more than a negligible advantage over a random guess. The group signature scheme is fully anonymous if no efficient adversary can have a strategy for winning the game.*

Definition 2. *Full traceability (cf [5]): The game is played by an adversary, also in two stages. In the first (choose) stage the adversary is given access to the second group managers’ secret key (the signature opening key) and can adaptively corrupt as many group members as it wishes. Let C be the set of corrupted*

members at the end of the first stage. State information (including the secret keys of the members of \mathcal{C}) is used as input to the guess stage, during which the adversary attempts to produce a message m and a valid group signature σ on m , such that if the (uncorrupted) OPEN protocol is invoked on (m, σ) , it will fail to attribute σ to any group member in the set \mathcal{C} . (Either the OPEN protocol would fail to produce a valid group member identity, or it would produce the identity of a member that has not been corrupted by the adversary.) The group signature scheme is said to be fully traceable if no efficient adversary can succeed in this game with non-negligible probability.

Remark 1. We also require that the compromise of either/both of the keys does not permit one to misattribute a signature issued by a legitimate group member. (Enrolled before the keys are compromised.) This means in particular that a group signature scheme is *not* a key escrow mechanism. This approaches differ from the one taken in [5]. There, it is the case that the first group manager escrows the users' secret keys – in particular users can be framed by compromising the first manager's secret key, which is equivalent to compromising *all* users' secret keys.

3 Preliminaries

In the group authentication problem a holder U of a group certificate interacts with a verifier V to prove his status as a group member without revealing his certificate. If the interactive protocol can be made non-interactive through the Fiat-Shamir heuristic ([20]), then the resulting algorithm will be similar to the issuing of a group signature, except that U 's identity may be unrecoverable from the signature alone. The issuing of a group signature requires, in addition to a proof of membership, that U *verifiably encrypts* some information about his certificate under the group manager's public key. U must provide the verifier with an encrypted token and prove to V that the group manager is able to decrypt the token to reveal U 's authorship of the signature.

A group signature can be seen as a proof of knowledge of a group certificate which provides evidence of membership. The group certificate can be generated only by the group manager GM and should be difficult to forge. In other words, the group membership certificate has the effect of a signature issued by the group manager. In addition, it has to contain some secret information generated by the group member and unknown to GM to avoid framing attacks in which GM signs on behalf of other members.

3.1 Modified ElGamal Signatures

Nyberg-Rueppel signatures [25] are ElGamal-type signature variants originally designed to provide message recovery. Instead of a one-way hash function, message-recovery schemes use a redundancy function. The redundancy function R is an one-to-one mapping of messages into a so-called message-signing space \mathcal{M}_S .

The image of R , denoted \mathcal{M}_R , must be sparse within \mathcal{M}_S i.e., given a random element of \mathcal{M}_S , there is a negligible probability of it being in \mathcal{M}_R . Otherwise, the message-recovery scheme is vulnerable to existential forgery attacks, as redundancy functions are, by definition, efficiently invertible. The following table assumes that $\mathcal{M}_S = \mathbf{Z}_p^*$. Again, the signature calls for a random input k , and the output is a pair (r, s) , where $r = R(m)g^{-k} \bmod p$, and s is computed as indicated in table 1.

Table 1. Nyberg-Rueppel signature variants.

Variant	Signing equation	Message recovery (verification)
I	$s = k^{-1}(1 + xr) \bmod q$	$R(m) = ry^{rs^{-1}}g^{s^{-1}} \bmod p$
II	$s = x^{-1}(-1 + kr) \bmod q$	$R(m) = ry^{sr^{-1}}g^{r^{-1}} \bmod p$
III	$s = -xr + k \bmod q$	$R(m) = ry^r g^s \bmod p$
IV	$s = -x + kr \bmod q$	$R(m) = ry^{r^{-1}}g^{sr^{-1}} \bmod p$
V	$s = x^{-1}(-r + k) \bmod q$	$R(m) = ry^s g^r \bmod p$
VI	$s = k^{-1}(x + r) \bmod q$	$R(m) = ry^{s^{-1}}g^{s^{-1}r} \bmod p$

If in the equations above, the redundancy function $R(\cdot)$ is replaced by an one-way function then the message-recovery property is lost. On the other hand, the requirement that the image of the function be sparse in the signing space may also be dropped. This modified Nyberg-Rueppel scheme, as a signature scheme of *short messages only*, is (loosely) reducible to the hardness of discrete logarithm computations in the standard model. Alternatively, it is (loosely) reducible to the discrete logarithm in the random oracle model if extended to arbitrarily long messages through the hash-and-sign paradigm. Moreover, the form of the modified verification equation – if the one-way function is suitably chosen – lends itself to the construction of proofs of knowledge of signatures that are more efficient. (When compared to similar proofs for unmodified ElGamal-type signature variants.)

We now describe the setting of our scheme. Let \mathcal{G} be some arithmetic group. Not all groups \mathcal{G} where Nyberg-Rueppel (or ElGamal) signatures make sense have the characteristics needed by our scheme. In section §4, we outline the specifics of the protocols in a suitable group, namely the subgroup of quadratic residues modulo a prime p , where p is simultaneously a *safe* prime, i.e. $p = 2q + 1$, with q also prime, and a *Sophie Germain* prime, that is the number $\hat{p} = 2p + 1$ is prime. There are other choices for the group \mathcal{G} , see appendix §C for a simpler construction in certain RSA rings.

Let \mathcal{G} be a suitable group. The order of \mathcal{G} may be a known prime or unknown composite number. Let g and g_1 be fixed, public generators for \mathcal{G} ; it is assumed that the discrete logarithm of g with respect to g_1 (and of g_1 w.r.t. g) is unknown to group members. Let $y = g^x$ be the public key of the signer GM ,

with associated secret x . (In the group signature scheme, y corresponds to the certificate issuing key.) Finally, this signature scheme defines the message space \mathcal{M} as the set of integers modulo q in the case of known order, and the set of integers smaller than some upper bound otherwise. The signing space is $\mathcal{M}_S = \mathcal{G}$, and let the one-way function $h(\cdot) : \mathcal{M} \rightarrow \mathcal{M}_S$ be defined by $h(m) = g_1^m$. Clearly, $h(\cdot)$ satisfies the requirements of a secure one-way function: $h(\cdot)$ is pre-image resistant by the hardness of computing discrete logarithms in \mathcal{G} . In the case of known order, it is further one-to-one, hence trivially collision-resistant. In the case of unknown order, finding a collision would reveal the order of \mathcal{G} , i.e., it is equivalent to factorization.

The signing and verification algorithms of the modified Nyberg-Rueppel are as follows:

$$\text{Signing:} \quad r = g_1^m g^{-k} \text{ (in } \mathcal{G}\text{);} \quad (1)$$

$$s = -xr + k \pmod{q}; \quad (2)$$

$$\text{Verification: } g_1^m = ry^r g^s \text{ (in } \mathcal{G}\text{).} \quad (3)$$

We have placed “mod q ” within parenthesis as that reduction is only computed when the order of \mathcal{G} is a known prime. These signatures are issuable only by the signer GM , who is privy to the secret key x associated to y . Indeed, such signatures are loosely reducible, through a standard forking lemma argument [26], to the discrete logarithm problem. Please refer to appendix §B.

3.2 High Level Description of the Scheme

A prospective new member U who wishes to join the group must have first secured a digital signature certificate with some certification authority. U starts the join protocol by choosing a random, secret value u and computing $I_U = g_1^u$. More precisely, U and GM interact so that both contribute to the randomization of u , while its value remains secret from the GM . Then U constructs a zero-knowledge proof (of knowledge) of the discrete logarithm of the pseudonym I_U with respect to g_1 . U signs the pseudonym and the proof of knowledge of the pseudonym secret, and sends it to the GM to request a group membership certificate.

GM verifies the signature against U 's public certificate and the correctness of the zero-knowledge proof. If both are well-formed, GM responds with the signature pair (r, s) on I_U , which is technically GM 's signature on a message u known only to U . This is safe from the GM 's viewpoint because both GM and U contribute to the choice of the value u . It is imperative, however, that only U knows the value u , as it is in effect the secret key allowing U to use the membership certificate to issue signatures. The equations used by GM to generate (r, s) are:

$$r = I_U g^{-k} \text{ (in } \mathcal{G}\text{); } s = -xr + k \pmod{q}, \quad (4)$$

where k is a random parameter of GM 's choice, and the reduction modulo q is applied only in the case of known order. U verifies the signature, checking that:

$$I_U = ry^r g^s \text{ (in } \mathcal{G}\text{).} \quad (5)$$

The scheme must permit U to prove knowledge of this certificate pair (r, s) without revealing any linkable function of r , s , or u . It must also allow GM to *open* the proof and show the identity of the group member. Both problems can be solved by employing a *verifiable encryption* of digital signature schemes. However, unlinkability between different protocol executions is not a requirement of verifiable encryption schemes, and indeed existing protocols for ElGamal-type signature schemes do not provide it. Hence, it would be possible to link two or more verifiable encryptions, which is equivalent to linking two or more group signatures from the same signer. This is because, in existing schemes, the first value r of the signature pair (r, s) is revealed and the actual protocol is applied only to the second value s , reducing then the problem of verifiable encryption of a digital signature to the simpler problem of verifiably encrypting a discrete logarithm (see [8,1,22,2] for details).

To solve this issue, it is necessary to ElGamal encrypt the value r as well, and prove in zero-knowledge that a Nyberg-Rueppel signature is known on a secret value u . More concretely, every time the group member must use the certificate, she encrypts the inverse of the value r , to get the ElGamal pair $(R_1, R_2) = (r^{-1}y_2^\ell, g_2^\ell)$. This encryption is under the second public key $y_2 = g_2^z$ of the group manager, used for opening group member signatures, with associated secret z .

The group member also encrypts his pseudonym: $(Y_1, Y_2) = (I_U y_2^{\ell'}, g_2^{\ell'})$. Notice that the product cipher is:

$$(R_1 Y_1, R_2 Y_2) = (I_U r^{-1} y_2^{\ell+\ell'}, g_2^{\ell+\ell'}) = (y^r g^s y_2^{\ell+\ell'}, g_2^{\ell+\ell'}) \quad (6)$$

In order to prove knowledge of a membership certificate, the member U releases the above ElGamal encrypted pairs (R_1, R_2) and (Y_1, Y_2) and proves that the product cipher encrypts some information which the signer can write in two ways, i.e., as the product $I_U r^{-1}$ for pseudonym I_U (for which the signer knows the corresponding pseudonym secret) and value r , and also as $y^r g^s$, for the same value r and some s known to the signer. In other words, the signer shows that an equation like (6) holds for the product cipher.

To proceed, we must overcome a difficulty with equation (6): The value in the exponent is reduced modulo the order of the group \mathcal{G} , while the encrypted value r is an element of \mathcal{G} itself. The reduction function does not preserve group operations, it is not multiplicative; and the method for proving equality between an ElGamal-encrypted value and a logarithm, due to Stadler [28], cannot be directly applied. The solution is to employ a technique due to Boudot [7] that permits efficient comparison between logarithms in different groups. So we use an auxiliary group \mathcal{F} of order compatible with the operations in \mathcal{G} . We release a commitment to the value r as an exponent of an element of \mathcal{F} , and we show that it equals (up to modular reduction), the exponent of y in the representation with respect to the basis $\{y, g\}$ of the value ElGamal encrypted in the product cipher $(R_1 Y_1, R_2 Y_2)$. Next, we use Stadler's technique to prove the equality of the encrypted value r (in the pair R_1, R_2 of \mathcal{G}), with the value committed as an exponent in \mathcal{F} .

To complete the sign protocol, the signer proves knowledge of the discrete logarithm to basis g of the value I_U which is ElGamal encrypted in the pair (Y_1, Y_2) . This shows that the group manager will be able to open the signature with just an ElGamal decryption operation.

Proofs of Knowledge. In this paper we make use of several types of proofs of knowledge about various relations between secrets. All these proofs of knowledge have been presented elsewhere. In order to harmonize the notation, which varies from author to author, and make the paper self-contained, we include an appendix (§A) in which we reproduce these various results.

4 The Scheme

We now describe the scheme more concretely, starting with \mathcal{T} , the set of shared public parameters. \mathcal{T} specifies security parameters δ , ϵ , σ , σ_2 , and τ , and a secure hash function \mathcal{H} that maps bit-strings of arbitrary length into bit-strings of fixed length τ . A typical set of choices would be $\delta = 40$, $\sigma = 40$, $\sigma_2 = 552$, $\tau = 160$, and $\mathcal{H}(\cdot) = \text{SHA-1}(\cdot)$. The parameter ϵ should be larger than 1 by a non-negligible amount. These security parameters impact the security and efficiency of the various proofs of knowledge used in the scheme. (Notation as in appendix §A.) \mathcal{T} also specifies an arithmetic group \mathcal{G} and three generators g , g_1 and g_2 of \mathcal{G} .

In this section we assume that \mathcal{G} is the quadratic residues subgroup of the multiplicative residues module p , where p is simultaneously a safe prime, i.e., and $p = 2q + 1$, with q also prime, and a Sophie Germain prime, i.e., the number $\hat{p} = 2p + 1$ is prime. Primes \hat{p} such that $\hat{p} = 2p + 1$, and $p = 2q + 1$, with p and q also prime are called *strong* primes. (More generally, if $\hat{p} = mp + 1$ and $p = nq + 1$ with small m , and n , are also called strong primes, but $m = n = 2$ gives the most efficient scheme.) See [18,21] for efficient methods to generate such primes. In order to choose g it is enough to pick a random element g' in \mathbf{Z}_p^* and set $g \equiv g'^2 \pmod{p}$, provided that $g \not\equiv 1 \pmod{p}$. The same procedure should be used to obtain g_1 and g_2 .

The scheme also requires an auxiliary group \mathcal{F} of order p , which in this section will be chosen as the quadratic subgroup of the multiplicative residues modulo \hat{p} . Furthermore, the scheme requires a second auxiliary group \mathcal{E} of unknown composite order \hat{n} . A trusted party generates a composite modulus n , plus a proof P that n is the product of two safe primes. The group \mathcal{E} is defined as the quadratic residue subgroup of the multiplicative residues modulo n . The order of \mathcal{E} is the universally unknown number $\phi(n)/4$. Group managers of competing organizations may all share the same modulus n , as the operation of the scheme does not require *anybody* to know the RSA trapdoor associated to n , and the trusted party may safely forget the factorization at its discretion.

The above public parameters can be further certified if so desired. A proof of primality can be provided for each of the primes; as for g , g_1 and g_2 , anybody can verify their correct generation by testing that each is not congruent to 0 or 1

Table 2. Shared and group specific parameters.

Shared parameters	
Security parameters: $\delta, \epsilon, \sigma, \sigma_2, \tau$;	
Secure hash function: $\mathcal{H}(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^\tau$;	
\hat{p}, p, q , primes s.t. $\hat{p} = 2p + 1$ and $p = 2q + 1$;	
$\mathcal{G} = \{x \in \mathbf{Z}_p^* : \exists a \in \mathbf{Z}_p^* \text{ s.t. } x \equiv a^2 \pmod{p}\}$;	
$\mathcal{F} = \{x \in \mathbf{Z}_{\hat{p}}^* : \exists a \in \mathbf{Z}_{\hat{p}}^* \text{ s.t. } x \equiv a^2 \pmod{\hat{p}}\}$;	
$\mathcal{E} = \{x \in \mathbf{Z}_n^* : \exists a \in \mathbf{Z}_n^* \text{ s.t. } x \equiv a^2 \pmod{n}\}$;	
g, g_1 , and g_2 , generators of \mathcal{G} .	
Group-specific parameters	
\mathcal{S} , a string including y and y_2 ;	
CA's signature: $\text{CERT}_{CA}(\mathcal{S})$.	

Table 3. The JOIN protocol

$U \rightarrow GM : J_U = I^m \pmod{p}$
$GM \rightarrow U : a, b \pmod{q}$
$U \rightarrow GM : \text{Sig}_U(I_U = J_U^a g_1^b, PK[u : I_U = g_1^u])$
$GM \rightarrow U : r = I_U g^{-k} \pmod{p}, s = -xr + k \pmod{q}$

modulo p , and then verifying that each is a square, by computing the Legendre symbol and checking that: $\left(\frac{q}{p}\right) = \left(\frac{q_1}{p}\right) = \left(\frac{q_2}{p}\right) = 1$.

In order to setup a group using the shared parameters above, the group manager GM chooses x and z at random among the numbers $[1, q - 1]$ and set the public keys $y = g^x$, and $y_2 = g_2^z$. The group manager should proceed to register these group-specific parameters with some certification authority. The GM would prepare a statement \mathcal{S} containing (minimally) a description of the group signature algorithms, a reference to the shared parameters, GM 's name, the group-specific parameters y, y_1 , and y_2 , and some timed information, such as start and expiration dates. The GM should obtain a certificate $\text{CERT}_{CA}(\mathcal{S})$ from the CA establishing the group-specific parameters.

Let $\text{Sig}_U(\cdot)$ denote U 's signature algorithm. To join the group, a prospective member U chooses a random secret m in the interval $[1, q - 1]$, computes $J_U = g_1^m$, and sends this value to GM , who responds with two values a , and b in $[1, q - 1]$. U computes his pseudonym as $I_U = J_U^a g_1^b$, and its associated secret $u = am + b \pmod{q}$. Next, U constructs a non-interactive proof of knowledge of the logarithm to basis g_1 of this pseudonym (see appendix A), and also his signature $S = \text{Sig}_U(I_U, PK)$ on both the pseudonym and the proof-of-knowledge just constructed. U forwards to the GM this signature S .

The GM now verifies that the pseudonym incorporated his contribution, i.e., $I_U = J_U^a g_1^b$. This step is important because u is unknown to GM , who must sign it. Since the GM contributed to u 's randomness, that does not constitute a threat to the GM 's signature algorithm. The GM also verifies the correctness of the proof-of-knowledge and U 's signature. If satisfied, the GM generates a

random $k \bmod q$, and computes $r = I_U g^{-k} \bmod p$, checking that $r < c$, where c equals:

$$c = p - 2^{\sigma+\tau/2+2} \sqrt{p}, \quad (7)$$

and repeating the process of computing other random k and r until such an r is found. Note that $r < c$ with overwhelming probability in a single attempt, because since the quadratic residues are nearly uniformly distributed in the interval $[1, p-1]$, we have that $r < c$ with probability close to $1 - \frac{2^{\sigma+\tau/2+2}}{\sqrt{p}} > 1 - 2^{-645}$ if the security parameters have the typical values $\delta = 40$, $\tau = 160$ and p has at least 768 significant bits. This very minor restriction on the possible values of r reflects requirements of the proof of equality of discrete logarithms in distinct groups, as we shall see later. After a suitable r is found, U computes $s = k - xr \bmod q$, and sends the certificate (r, s) to U . The GM also records the signature S , which ties U 's identity to the certificate's pseudonym. U verifies that the certificate (r, s) satisfies the verification equation, and if so, accepts it as valid.

We now describe the protocol **SIGN**. One goal of this protocol is that U convince a verifier V of its knowledge of a membership certificate (r, s) as above. As in section §3, the signer chooses random ℓ , and ℓ' , with $0 < \ell, \ell' < q$. U releases the ElGamal encrypted pairs:

$$(Y_1, Y_2) = (I_U y_2^{\ell'}, g_2^{\ell'}); \quad (R_1, R_2) = (r^{-1} y_2^{\ell}, g_2^{\ell});$$

Next, U demonstrates that the pseudonym I_U is encrypted by the pair (Y_1, Y_2) , and proves knowledge of the pseudonym secret u , by executing $PK[u, \ell' : Y_1 = g_1^u y_2^{\ell'} \wedge Y_2 = g_2^{\ell'}]$. This step is crucial to prevent framing attacks against U , as not even the group manager can execute it without knowledge of u .

Continuing with the **SIGN** protocol, U generates a fresh, random generator χ of the group \mathcal{F} , and computes a (computationally zero-knowledge) commitment to the value r as $E_1 = E_1(r, 0) = \chi^r$. In the language of appendix §A, this is a (degenerate) commitment to the value r in the group \mathcal{F} , with respect to the generator χ .

U also generates a commitment to r in the auxiliary group \mathcal{E} of unknown order. For that, U uses two generators β and γ of \mathcal{E} , where β and γ are provably randomly generated, so that U cannot know their relative discrete logarithm. For instance, γ and β can be generated as the squares of two consecutive values of a secure pseudo-random number generator *SPRNG*. The commitment is computed as $E_2 = E_2(r, s_2) = \gamma^r \beta^{s_2}$, where s_2 is a random parameter of U 's choice: $s_2 \in [-2^{\kappa+\tau+1}, 2^{\kappa+\tau+1}]$, where $2^{\kappa-1} \leq |\mathcal{E}| < 2^\kappa$. Notice that the value $R_1 Y_1 = I_U r^{-1} y_2^{\ell+\ell'} = y^r g^s y_2^{\ell+\ell'}$ is also a commitment to the value r in the group \mathcal{G} , with generators y, g , and y_2 . Denote it by $E_3 = R_1 Y_1$.

In the next step, U reveals the commitments E_1, E_2 , and the respective generators γ, β , and χ . (In the case of γ and β , U must also reveal the seed of the *SPRNG* that leads to the computation of γ and β .) U then shows that E_1, E_2 and E_3 all are commitments to the same value r . (Notice that we are following the efficient construction found in [7], repeated in detail here

for reasons of convenience.) U executes two proofs of equality of two committed values (def. 10). In the first proof U sends V a triple (c', D', D'_1) satisfying: $c' = \mathcal{H}(\chi || \gamma || \beta || E_1 || E_2 || \chi^{D'} E_1^{-c'} \bmod \hat{p} || \gamma^{D'} \beta^{D'_1} E_2^{-c'} \bmod n)$. Again, refer to def. (10) for how to build these proofs. In agreement with the notation in appendix §A, we denote the above by $PK[r, s_2 : E_1 = E_1(r, 0) \wedge E_2 = E_2(r, s_2)]$. Then U sends V a quintuple (c, D, D_1, D_2, D_3) satisfying: $c = \mathcal{H}(\gamma || \beta || y || g || y_2 || E_2 || E_3 || \gamma^D \beta^{D_1} E_2^{-c} \bmod n || y^D g^{D_2} y_2^{D_3} E_3^{-c} \bmod p || g_2^{D_3} (Y_2 R_2)^{-c} \bmod p)$. Denote that by $PK[r, s, s_2, t : E_2 = E_2(r, s_2) \wedge E_3 = E_3(r, s, t) \wedge Y_2 R_2 = g_2^t]$.

If all of the commitments E_1 , E_2 , and E_3 took place within the same group the above would be a proof of equality of the committed exponent in each of the commitments. However, as the order of the groups differ, we have only proved knowledge of an integer value r which satisfies

$$r \equiv r_1 \pmod{p}, \text{ and } r \equiv r_3 \pmod{q}, \quad (8)$$

where r_1 and r_3 are, respectively, the exponents committed in E_1 and E_3 , while r is the exponent committed in E_2 . (As U does not know the order of \mathcal{E} , it cannot set up a modular equation that the exponent of E_2 should satisfy, and must use the full integer value r .) U could cheat and pass the “proof” above for any two different values r_1 and r_3 , by setting r in E_2 to equal the solution, computed via the Chinese Remainder Theorem, to the pair of modular equations in (8). Thus, a non-member U' would be able to forge the proof of knowledge of a certificate, by choosing r_3 and s arbitrarily, computing the value r_1 that would make the certificate equation work, and then solving the pair of equations (8) for an r that reduces to $r_1 \pmod{p}$ and $r_3 \pmod{q}$, respectively. In the cheating case, however, because $r_1 \not\equiv r_3 \pmod{q}$, U' computes a value $r > p$ as the solution of 8. Thus, if U' is required to prove that the value r_2 committed in E_2 is within an interval of width at most p , this forgery attack is prevented; and the commitments must all hide the same value. So to complete the “proof of equality of commitments in different groups,” U must construct a proof that the value r is restricted to an interval of width at most p . For that, U uses the fact that $r < c$, and constructs the proof of knowledge that a committed value lies in a slightly larger interval, def. (13): $PK[r, s_2 : E_2 = E_2(r, s_2) \wedge r \in [-2^{\delta+\tau/2+1}\sqrt{c}, c + 2^{\delta+\tau/2+1}\sqrt{c}]]$. To observe that the interval in question has width smaller than p , notice that its width equals $c + 2^{\delta+\tau/2+2}\sqrt{c} < c + 2^{\delta+\tau/2+2}\sqrt{p} = p$, by choice of c (see equation 7).

Finally, U must show that the exponent committed in E_1 equals the value encrypted in the pair (R_1, R_2) , by executing (definition 14): $PK[r, t : E_1 = \chi^r \wedge R_1 = r^{-1}y_2^t \wedge W_2 = g_2^t]$. The actual protocol SIGN combines all the proofs of knowledge into a single signature of knowledge. This is done by simultaneously committing to all the inputs of the proofs and using the resulting challenge in all the verification equations (à la Fiat-Shamir). In addition, the message \mathcal{M} to be signed is used as an extra input of the hash function.

The protocol is summarized in table 4. Moreover, algorithm VERIFY can be derived immediately from the above formal description of SIGN as a proof of knowledge of a group certificate.

Table 4. The SIGN protocol

<u>Proof arguments:</u>
$Y_1, Y_2, R_1, R_2, \chi, \gamma, \beta, E_1, \text{ and } E_2.$
<u>Signature of knowledge:</u>
$SPK[u, \ell', \ell, r, s, s_2, t : Y_1 = g_1^u y_2^{\ell'} \wedge Y_2 = g_2^{\ell'} \wedge E_1 = E_1(r, 0) = \chi^r \wedge R_1 = r^{-1} y_2^\ell \wedge R_2 = g_2^\ell$
$\wedge E_2 = E_2(r, s_2) = \gamma^r \beta^{s_2} \wedge r \in [-2^{\delta+\tau/2+1}\sqrt{c}, c + 2^{\delta+\tau/2+1}\sqrt{c}]$
$\wedge E_3 = E_3(r, s, t) = Y_1 R_1 = y^r g^s y_2^t \wedge Y_2 R_2 = g_2^t](\mathcal{M})$

As for OPEN, it is enough that the group manager decrypts the pair (Y_1, Y_2) to obtain the value I_U and the corresponding group membership certificate. GM constructs a proof that I_U is indeed the value encrypted in (Y_1, Y_2) *without revealing the group secret x* : $PK[x : Y_1 I_U^{-1} = Y_2^x \wedge y_2 = g_2^x]$, a publicly verifiable *proof of authorship* of the signature.

5 Conclusions

In this paper we introduced the first group signature scheme with constant-size parameters that does not require any group members, including group managers, to know trapdoor secrets. Our scheme is not bound to a specific setting but it can work in various groups where the Decision Diffie-Hellman assumption holds: The appendix §C contains a simpler construction in an RSA ring.

Our scheme is less efficient than the state-of-the-art scheme in [3]. However, the scheme in [3] requires the group manager to know trapdoor information which cannot be shared with other group managers, thus making it difficult to enable collaboration among distinct groups.

Acknowledgments

We are grateful to an anonymous referee who suggested changes to our security arguments that eventually led to a simplification of the scheme and its proof of security.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Proc. of EUROCRYPT '98*, LNCS vol. 1403, Springer-Verlag, 1998.
2. G. Ateniese. Efficient Verifiable Encryption (and Fair Exchange) of Digital Signatures. In 6th ACM CCS, pp. 138-146, 1999.
3. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Proc. of CRYPTO 2000*, LNCS, Springer-Verlag, 2000.

4. G. Ateniese and G. Tsudik. Some open issues and directions in group signatures. In *Financial Cryptography* 1999. LNCS 1648, Springer-Verlag, 1999.
5. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction based on General Assumptions. In *Proc. of EUROCRYPT '03*, LNCS vol. 2656, Springer-Verlag, 2003.
6. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of the 1st ACM CCS*, 1993.
7. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Proc. of EUROCRYPT'00*, LNCS vol. 1807, Springer Verlag, 2000.
8. Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *Proc. of ASIACRYPT 2000*, LNCS vol. 1976, Springer Verlag.
9. Jan Camenisch and Anna Lysyanskaya. Efficient Non-transferable Anonymous Multi-show Credential System with Optional Anonymity Revocation. In *Proc. of EUROCRYPT'01*, Springer Verlag, 2001.
10. Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency. In *Proc. of ASIACRYPT '98*, LNCS vol. 1514, Springer-Verlag, 1998.
11. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Proc. of CRYPTO'97*, LNCS vol. 1296, Springer-Verlag, 1997.
12. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. In *Proc. of EUROCRYPT'98*, LNCS vol. 1403, Springer-Verlag, 1998.
13. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. Updated version with corrections, GTE Technical Report. 1998. Available at <http://www.ccs.neu.edu/home/yiannis>.
14. D. Chaum, *Security Without Identification: Transactions Systems to Make Big Brother Obsolete*, CACM Vol. 28, No. 10, October 1985.
15. D. Chaum and J. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *Proc. of CRYPTO'86*, pp. 118-167, Springer-Verlag, 1986.
16. D. Chaum and E. van Heyst. Group signatures. In *Proc. of CRYPTO'91*, LNCS vol. 547, Springer-Verlag, 1991.
17. L. Chen. Access with pseudonyms. In *Cryptography: Policy and Algorithms*, pp. 232-243, Springer-Verlag, 1995.
18. Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In *ACM Transactions on Information and System Security*, 2000.
19. I. Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In *Proc. of CRYPTO '88*, Springer-Verlag, 1988.
20. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. of CRYPTO'86*, Springer Verlag, 1987.
21. Marc Joye, Pascal Paillier and Serge Vaudenay. Efficient generation of prime numbers. In *Cryptographic Hardware and Embedded Systems – CHES 2000*, LNCS vol. 1965, Springer Verlag, 2000.
22. J. Kilian and E. Petrank. Identity escrow. In *CRYPTO '98*, vol.1642 of LNCS, pp. 169-185, Berlin, 1998. Springer-Verlag.
23. A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym Systems. In *Selected Areas in Cryptography*. Springer-Verlag 1999.
24. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. Chapter §11, note 11.83, pp. 461–462. CRC Press, 1996.

25. Kaisa Nyberg and Rainer A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. In *Proc. of EUROCRYPT'94*, Springer Verlag, 1994.
26. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Proc. of EUROCRYPT'96*, Springer Verlag, newblock 1996.
27. Amit Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In *Proc. of the 40th FOCS Symposium*. 1999.
28. Markus Stadler. Publicly Verifiable Secret Sharing. In *Proc. of EUROCRYPT'96*, LNCS vol. 1070, Springer Verlag, 1996.

A Proofs of Knowledge

All the proofs of knowledge listed in this section have been proved zero-knowledge in a statistical or computational sense within the random oracle model, under the Decisional Diffie-Hellman assumption, and the Strong RSA assumption, explained below.

Notation 1 (Groups and generators).

- \mathcal{J} stands for an arithmetic group, such as an RSA ring with composite modulus n or the group \mathbf{Z}_p^* of non-zero (multiplicative) residues modulo p .
- g stands for an element of \mathcal{J} of unknown composite order or known prime order. Let q be the order of g .
- Let κ be the smallest integer such that 2^κ is larger than q . We assume that κ is known, even if q is not.
- g generates the subgroup \mathcal{G} of \mathcal{J} .

Let \mathcal{H} stand for a secure hash function which maps arbitrarily long bit-strings into bit-strings of fixed length τ . Let ϵ denote a second security parameter.

Definition 3 (Decisional Diffie-Hellman assumption (DDH)). *Let \mathcal{J} be a group and g an element of known prime, or unknown composite, order q in \mathcal{J} . Let $\mathcal{G} = \langle g \rangle$ be the subgroup generated by g in \mathcal{J} . The DDH assumption for \mathcal{G} is then there is no efficient (randomized, probabilistic) algorithm that can distinguish between the two following distributions in \mathcal{G} :*

$$\{(h, i, j), \text{ where } h, i, j \text{ are independently randomly distributed (i.r.d.) in } \mathcal{G}\}$$

and

$$\{(h', i', j'), \text{ where } h' = g^x, i' = g^y, j' = g^{xy} \text{ for i.r.d. } x, y \text{ with } 0 \leq x, y < q\}$$

A triple of group elements such as (h', i', j') above is called a *Diffie-Hellman triple*. The DDH assumption is thus the statement that there is no efficient algorithm to distinguish between Diffie-Hellman triples and randomly generated triples.

Definition 4 (Strong RSA assumption (SRSA)). Let $n = pq$ be a composite modulus, where p and q are two large primes. The strong RSA assumption states that there is no efficient (randomized, probabilistic) algorithm that, given as input n and an integer y , but not the factorization of n , can produce two other integers u and e , where $e > 1$ and $u^e \equiv y \pmod{n}$.

SRSA underlies the security of the proof of equality of logarithms in distinct groups (10).

Definition 5 (Proof of knowledge of a discrete logarithm). U can prove to a verifier V his knowledge of an integer x in $\{0, \dots, 2^\kappa - 1\}$, such that $h = g^x$, by releasing integers s and c , with s in $\{-2^{\epsilon(\tau+\kappa)+1}, \dots, 2^{\epsilon(\tau+\kappa)+1} - 1\}$ and c in $\{0, \dots, 2^\tau - 1\}$, s.t. $c = \mathcal{H}(g||h||g^s h^c)$, where the symbol $||$ denotes string concatenation.

In order to compute the pair (s, c) , U generates a random integer k in $\{-2^{\epsilon(\tau+\kappa)}, \dots, 2^{\epsilon(\tau+\kappa)} - 1\}$ and sets $c = \mathcal{H}(g||h||g^k)$, and $s = k - cx$ (as integer). Denote it by (notation introduced in [11]): $PK[x : h = g^x]$.

This proof of knowledge can be transformed into a digital signature, with x being the secret key associated with public key h . To sign an arbitrary bitstring m , we instead compute c as: $c = \mathcal{H}(g||h||g^s h^c||m)$. Denote this *signature of knowledge* ([11]) by: $SPK[x : h = g^x](m)$.

Returning to the notation in definition (5), if the order q of the group \mathcal{G} is known, then operations on the exponents should be computed modulo q , and some statements about the size of parameters can be simplified. In the above we would substitute:

$$\begin{aligned} x &\in \{0, \dots, 2^\kappa - 1\} \text{ by } x \in \{0, \dots, q - 1\}, \\ s &\in \{-2^{\epsilon(\tau+\kappa)+1}, \dots, 2^{\epsilon(\tau+\kappa)+1} - 1\} \text{ by } s \in \{0, \dots, q - 1\}, \text{ and} \\ s &= k - cx \text{ (in } \mathbf{Z}) \text{ by } s = k - cx \pmod{q}. \end{aligned}$$

In the following definitions we assume the group order q is unknown; as above, it is straightforward to adapt them to the case of known order.

Definition 6 (Proof of knowledge of a common discrete logarithm). U can prove to a verifier V his knowledge of an x (with $0 \leq x < 2^\kappa$) s.t. two lists g_1, g_2, \dots, g_ℓ and h_1, h_2, \dots, h_ℓ (of elements of \mathcal{G}) satisfy $h_i = g_i^x, i = 1 \dots \ell$, by releasing s and c ($-2^{\epsilon(\tau+\kappa)+1} \leq s < 2^{\epsilon(\tau+\kappa)+1}$ and $0 \leq c < 2^\tau$) s.t.

$$c = \mathcal{H}(g_1||\dots||g_\ell||h_1||\dots||h_\ell||(g_1 \dots g_\ell)^s (h_1 \dots h_\ell)^c).$$

U computes $c = \mathcal{H}(g_1||\dots||g_\ell||h_1||\dots||h_\ell||(g_1 \dots g_\ell)^k)$ for a randomly chosen k ($-2^{\epsilon(\tau+\kappa)} \leq k < 2^{\epsilon(\tau+\kappa)}$), and sets $s = k - cx$. Denote it by: $PK[x : h_1 = g_1^x \wedge \dots \wedge h_\ell = g_\ell^x]$.

Definition 7 (Proof of knowledge of a representation). U can prove his knowledge of elements x_1, \dots, x_ℓ (with $0 \leq x_i < 2^\kappa$) s.t. a given element A satisfies $A = g_1^{x_1} \dots g_\ell^{x_\ell}$, by releasing s_i and c ($-2^{\epsilon(\tau+\kappa)+1} \leq s_i < 2^{\epsilon(\tau+\kappa)+1}; 0 \leq c < 2^\tau$) s.t. $c = \mathcal{H}(g_1||\dots||g_\ell||A||g_1^{s_1} \dots g_\ell^{s_\ell} A^c)$.

Again, U computes $c = \mathcal{H}(g_1 || \dots || g_\ell || A || g_1^{k_1} \dots g_\ell^{k_\ell})$ for randomly chosen $k_i (-2^{\epsilon(\tau+\kappa)} \leq k_i < 2^{\epsilon(\tau+\kappa)})$, and sets $s_i = k_i - cx_i$. Denote it by: $PK[x_1, \dots, x_\ell : A = g_1^{x_1} \dots g_\ell^{x_\ell}]$.

The next two proofs of knowledge assert that a committed value lies in an interval. The first one was introduced in [12], and corrected in [13]. The second one, which uses the first as building block, was introduced in [7], and is used in our scheme.

Let g, h be two elements of \mathcal{G} . Assume that g and h are constructed in a provably random way, for instance as consecutive images of a secure pseudo-random generator. Generating g and h in such a way ensures that no one knows the discrete logarithm of g to basis h , or that of h to basis g .

Definition 8 (Commitment to a secret value). *Let x be a secret value held by U . Let g and h be two provably random generators of \mathcal{G} . We say that $E = E(x, r) = g^x h^r$ is a commitment to the value x in \mathcal{G} , where r is a randomly generated value, $0 < r < q$.*

If q is unknown, then one must choose r in a larger interval, say $-2^{\kappa+\tau+1} < r < 2^{\kappa+\tau+1}$, to ensure that all elements in the interval $[0, q-1]$ are sampled nearly uniformly. The commitment reveals nothing about r in a statistical sense.

Let \mathcal{E} be a distinct arithmetic group of unknown composite order n . For instance, \mathcal{E} can be chosen as the subgroup of quadratic residues in an RSA ring. Let $g = g_1, g_2, h = h_1$, and h_2 be provably random generators of \mathcal{E} . We assume that the smallest integer λ s.t. $2^\lambda > n$ is known. Assume U has published two commitments, $E = E_1(x, r) = g_1^x h_1^{r_1}$ in \mathcal{G} , and a second commitment $E_2(x, r_2) = g_2^x h_2^{r_2}$.

Let δ, σ and σ_2 be other security parameters. Assume further that $x < b$.

Definition 9 (Proof of knowledge of a committed value). *U can prove in ZK to a verifier V knowledge of a number x committed through $E = E(x, r) = g^x h^r$, by sending V a triple (c, D, D_1) satisfying: $c = \mathcal{H}(g || h || E || g^D h^{D_1} E^{-c} \bmod n)$.*

U generates random $t \in [1, 2^{\delta+\tau/2}b + 1]$ and $s \in [1, 2^{\delta+\tau/2+\sigma}n - 1]$; computes $W = g^t h^s \bmod n$; computes $c = \mathcal{H}(g || h || E || W)$; and finally computes $D = t + cx, D_1 = s + cr$ (in \mathbf{Z}).

Definition 10 (Proof of equality of two committed values). *U can prove in ZK to a verifier V that two commitments $E_1 = E_1(x, r_1)$ and $E_2 = E_2(x, r_2)$ hide the same exponent x , by sending V a quadruple (c, D, D_1, D_2) satisfying: $c = \mathcal{H}(g_1 || h_1 || g_2 || h_2 || E_1 || E_2 || g_1^{D_1} h_1^{D_1} E_1^{-c} \bmod n || g_2^{D_2} h_2^{D_2} E_2^{-c} \bmod n)$.*

U generates the random values $t \in [1, 2^{\delta+\tau/2}b + 1]$, $s_1 \in [1, 2^{\delta+\tau/2+\sigma}n - 1]$, and $s_2 \in [1, 2^{\delta+\tau/2+\sigma_2}n - 1]$. Next, U computes $W_1 = g_1^t h_1^{s_1} \bmod n$, $W_2 = g_2^t h_2^{s_2} \bmod n$; and sets $c = \mathcal{H}(g_1 || h_1 || g_2 || h_2 || E_1 || W_1 || W_2)$. Finally, U computes $D = t + cx, D_1 = s_1 + cr_1, D_2 = s_2 + cr_2$ (in \mathbf{Z}). Denote this by $PK[x, r_1, r_2 : E_1 = E_1(x, r_1) \wedge E_2 = E_2(x, r_2)]$.

Definition 11 (Proof that a committed number is a square). *U can convince a verifier V that the commitment $E = E(x^2, r_1) = g^{x^2} h^{r_1} \bmod n$ ($r_1 \in [-2^\sigma n + 1, 2^\sigma n - 1]$) contains the square of a number known to U , by sending V the quintuple (F, c, D, D_1, D_2) , where $c = \mathcal{H}(g||h||E||F||F^{D_1} h^{D_2} E^{-c} \bmod n || g^{D_2} h^{D_2} F^{-c} \bmod n)$.*

Indeed, U generates a random r_2 in $[-2^\sigma n + 1, 2^\sigma n - 1]$, and sets $F = g^x h^{r_2}$. Notice now that U can rewrite E in the basis $\{F, h\}$ as $E(x, r^3) = F^x h^{r_3} \bmod n$, where $r_3 = r_1 - r_2 x$, and $r_3 \in [-2^\sigma b n + 1, 2^\sigma b n - 1]$. It is enough then for U to use the previous proof of equality of the exponent x committed though $E_1 = F = E(x, r_2)$ and $E_2 = E = E(x, r_3)$, i.e., execute $PK[x, r_2, r_3 : F = g^x h^{r_2} \wedge E = F^x h^{r_3}]$. Denote this by $PK[x, r_1 : E = E(x^2, r_1)]$.

Definition 12 (Proof that a committed number lies in a larger interval). *A prover U can convince a verifier V that a number $x \in [0, b]$ which is committed in $E = E(x, r) = g^x h^r \bmod n$ ($r \in [-2^\sigma n + 1, 2^\sigma n - 1]$), lies in the much larger interval $[-2^{\delta+\tau/2} b, 2^{\delta+\tau/2} b]$, by sending V the triple (C, D_1, D_2) , where $D_1 \in [cb, 2^{\delta+\tau/2} b - 1]$, and $C = \mathcal{H}(g||h||E||g^{D_1} h^{D_2} E^{-c}; c = C \bmod 2^{\tau/2})$.*

U generates randoms $s \in [0, 2^{\delta+\tau/2} b - 1]$, $t \in [-2^{\delta+\tau/2+\sigma} n + 1, 2^{\delta+\tau/2+\sigma} n - 1]$; computes $W = g^s h^t \bmod n$; computes $C = \mathcal{H}(g||h||E||W)$, and $c = C \bmod 2^{\tau/2}$; and sets $D_1 = s + cx$, $D_2 = t + cr$, repeating the procedure from the beginning if $D_1 \notin [cb, 2^{\delta+\tau/2} b - 1]$. We denote the above by $PK_{CFT}[x, r : E = E(x, r) \wedge x \in [-2^{\delta+\tau/2} b, 2^{\delta+\tau/2} b]]$.

Definition 13 (Proof that a committed number lies in a slightly larger interval). *A prover U can convince a verifier V that a number $x \in [a, b]$, committed in $E = E(x, r) = g^x h^r \bmod n$ ($r \in [-2^\sigma n + 1, 2^\sigma n - 1]$) lies in the slightly larger interval $[a - \alpha, b + \alpha]$, where $\alpha = 2^{\delta+\tau/2+1} \sqrt{b - a}$, by releasing \tilde{E}_1, \bar{E}_1 , and proving: $PK[x, r : E = E(x, r)]$, $PK[\tilde{x}_1, \tilde{r}_1 : \tilde{E}_1 = E(\tilde{x}_1^2, \tilde{r}_1)]$, $PK[\bar{x}_1, \bar{r}_1 : \bar{E}_1 = E(\bar{x}_1^2, \bar{r}_1)]$, $PK_{CFT}[\tilde{x}_2, \tilde{r}_2 : \tilde{E}_2 = E(\tilde{x}_2, \tilde{r}_2) \wedge \tilde{x}_2 \in [-\alpha, \alpha]]$, where $\tilde{E}_2 = \frac{E}{g^a \bar{E}_1} \bmod n$, $PK_{CFT}[\bar{x}_2, \bar{r}_2 : \bar{E}_2 = E(\bar{x}_2, \bar{r}_2) \wedge \bar{x}_2 \in [-\alpha, \alpha]]$, where $\bar{E}_2 = \frac{g^b}{\bar{E}_1} \bmod n$.*

U computes $\tilde{E} = E/g^a \bmod n$, $\bar{E} = g^b/E \bmod n$; sets $\tilde{x} = x - a$ and $\bar{x} = b - x$; computes $\tilde{x}_1 = \lfloor \sqrt{x - a} \rfloor$, $\tilde{x}_2 = \tilde{x} - \tilde{x}_1^2$, $\bar{x}_1 = \lfloor \sqrt{b - x} \rfloor$, $\bar{x}_2 = \bar{x} - \bar{x}_1^2$; generates random \tilde{r}_1 and \tilde{r}_2 in $[-2^\sigma n + 1, 2^\sigma n - 1]$ s.t. $\tilde{r}_1 + \tilde{r}_2 = r$, and similarly \bar{r}_1, \bar{r}_2 s.t. $\bar{r}_1 + \bar{r}_2 = -r$; computes the commitments $\tilde{E}_1 = E(\tilde{x}_1^2, \tilde{r}_1)$, $\tilde{E}_2 = E(\tilde{x}_2, \tilde{r}_2)$, $\bar{E}_1 = E(\bar{x}_1^2, \bar{r}_1)$, and $\bar{E}_2 = E(\bar{x}_2, \bar{r}_2)$; and executes the proofs of knowledge listed in the above definition. We denote the above proof of knowledge by $PK[x, r : E = E(x, r) \wedge x \in [a - \alpha, b + \alpha]]$.

The last cryptographic building block we need is the verifiable ElGamal encryption of an exponent.

Definition 14 (Verifiable ElGamal encryption of an exponent). *Assume U holds a secret r , and has published the value $\omega = \chi^r$. Here χ is a generator of a group \mathcal{F} of order n , where n may be prime or composite, and $0 < r < n$. We*

assume that the DDH assumption holds in \mathcal{F} . It is possible for U to prove in zero-knowledge that a pair $(A = r^{-1}y^a, B = g^a) \bmod n$, is an ElGamal encryption under public key y of the exponent of ω to basis χ .

We denote it by: $PK[r : \omega = \chi^r \wedge A = r^{-1}y^a \wedge B = g^a]$. The proof can be found in [28], and we repeat it here for convenience. For i in $\{1, \dots, \nu\}$, U generates random t_i , and computes $g_i = g^{t_i}$, $y_i = y^{t_i}$, and $\omega_i = \chi^{y_i}$. Next, U computes

$$c = \mathcal{H}(\chi \parallel \omega \parallel A \parallel B \parallel g_1 \parallel \omega_1 \parallel \dots \parallel g_\nu \parallel \omega_\nu). \quad (9)$$

Next, U computes $s_i = t_i - c_i a$, where c_i stand for the i^{th} -bit of c . The proof consists of c and s_i , $i = 1, \dots, \nu$. In order to verify, V recomputes $g_i = g^{s_i} B^{c_i}$, $y'_i = y^{s_i} A^{c_i}$, and $\omega_i = \omega^{y'_i}$, and checks that (9) holds. The rationale for the proof is that, when $c_i = 0$, the verifier checks that g_i and ω_i are correctly constructed; when $c_i = 1$, the verifier checks that (A, B) is the ElGamal Encryption of the discrete logarithm of ω to basis χ , provided that g_i and ω_i are constructed correctly. If the statement were false, U could pass only one of the verification equations, for each i . In the random oracle model, the probability of U successfully proving a false statement is $2^{-\nu}$.

B Security Analysis

Before the introduction of a formal model of security of group signature schemes [5], it was common practice to prove the security of a scheme by showing that it would satisfy the various informal requirements listed in section §2. Of course, it is impossible to be sure that any such list is complete, and in fact early schemes failed to identify the need for resistance against coalition/collusion attacks (see [4] for a discussion about this issue).

Thanks to the formal model, a clearer picture about the complete security requirements of group signatures has now emerged; a scheme proven to satisfy “full anonymity” and “full traceability” can be trusted to provide security – at least as long as the particular computational assumptions underlying the cryptographic primitives (digital signatures, encryption, proofs-of-knowledge) used in the scheme hold up. Unfortunately it is challenging to provide a proof in the new model. The only example of such a proof is for the general construction given in [5] itself. While that construction shares similar design principles with ours, their proof works in a different model of computation. In particular, security conditions for the proofs-of-knowledge are defined in the Common Reference String model. On the other hand, the primitives used in our scheme are provably secure only in the Random Oracle Model (ROM). Indeed, ALL primitives based on discrete logarithms (which we must use if the scheme is to be functionally trapdoor-free) are only proven secure in the ROM model. Thus, in order to provide a formal security proof, we would have to adapt the framework of [5] to the ROM setting. We plan to pursue this direction in a future journal publication of this work. In this section we will give some arguments on how such a formal proof would work for our scheme. Before we proceed, however, we would like

to remark that it is simple to prove the security of our scheme by going over each property in §2. In fact, the only requirement that is not clear from the construction is security against coalition attacks. Equivalently, it is not obvious whether group membership certificates are unforgeable even if some (or all) the group members conspire to share their secrets, because our scheme uses a new, modified Nyberg-Rueppel signature for certificate issuance. Indeed, certificate unforgeability is equivalent to the property that this signature be existentially unforgeable under active attacks. We now prove the security of the modified Nyberg-Rueppel.

Proposition 1 (Forking lemma for modified Nyberg-Rueppel). *Let A be an adversary which attempts to forge modified Nyberg-Rueppel signatures on messages issued under the public key $y = g^x$. Assume A has a non-negligible probability of success, as computed over the sample space of messages m , random tapes r and random bases g_1 . Then A has a non-negligible probability of success of computing relative discrete logarithms in the group \mathcal{G} .*

Proof. Since A has non-negligible success probability over sample triples (m, r, g_1) , a standard product sample argument can be used to show that for a non-negligible set of choices of values for the first two components, (i.e., values for the message m and random tape r) the algorithm has a non-negligible probability of success over choices for the remaining component (the basis g_1 in \mathcal{G}). Now consider the following reduction to the relative discrete logarithm problem. Given two arbitrary values g_2 and g_3 in \mathcal{G} , choose (with non-negligible probability of success) values m and r such that A can forge signatures on message m with random tape r for a non-negligible subset of bases g_1 in \mathcal{G} . Then, with non-negligible probability, both g_2 and g_3 will belong to that subset. But this implies that A can compute a pair (m, r) and values s and s' such that $g_2^m = ry^r g^s$ and $g_3^m = ry^r g^{s'}$. Dividing the equations, we get $\left(\frac{g_2}{g_3}\right)^m = g^{s-s'}$, which implies $\text{dlog}_{g_3}(g_2) = \frac{s-s'}{m}$.

Proposition 2. *The modified Nyberg-Rueppel signature scheme, as a signature scheme on short messages, is existentially unforgeable under chosen message attacks, if the discrete logarithm problem is hard in \mathcal{G} .*

Proof. Since we are considering short messages only, there is no need to use the random oracle model. The previous proposition reduces such forgeries to the hardness of discrete logarithm computations. Of course the reduction is “loose” by a factor of 2: If you can forge signatures with probability at least p , the probability of successful computation of discrete logarithms is at least p^2 .

Notice that the SIGN protocol is a Schnorr-type signature scheme, in the sense that it binds all the signature parameters in a single hash computation, and the signer’s secret is a discrete logarithm. In fact, the signature itself includes a proof of knowledge of discrete logarithm of the signer’s public key with respect to a fixed basis (also tied in the hash computation). Such constructions can be

proven secure in the random oracle model [26]. In other words, individual group member signatures are secure against existential forgery by adaptively chosen message attacks.

Consider now the anonymity game. The attacker has corrupted all secret keys of all group members. It is allowed to query an OPEN oracle for opening arbitrary valid signatures. After possibly some interaction with the oracle it can choose two identities i_0 and i_1 and a message m . The adversary challenge σ is then a valid group signature on m that is known to have been issued by either i_0 or i_1 with equal probability. The adversary is allowed to further interact with the OPEN oracle, but is now restricted not to query the oracle with the challenge (m, σ) .

Claim (Reduction to passive attacks). Assume that the group member signature is secure against existential forgery by adaptively chosen message attacks, and that it implements a sound zero-knowledge proof of knowledge of a certificate on a pseudonym and its associated secret. If there is an efficient attacker that, upon interacting with an OPEN oracle, can guess the identity of the signer on the challenge with non-negligible advantage over a random guess, then there is an efficient attacker *without access to an OPEN oracle* that can similarly guess the identity of the signer with non-negligible advantage over a random guess.

Argument. The idea for the proof is as follows: Let A_0 be an attacker with access to the oracle, and A_1 an attacker that has full access to ALL the group members for all time – i.e., it is able to see the internal state of the group members that lead to computation of group signatures (except that he cannot see the computation of the challenge). However, A_1 is not given access to the oracle. Let Q be some query made by A_0 to the oracle. If the oracle accepts and decrypts the message, then it means that either the query included a valid group member signature or that the proof of knowledge was forged. Since we assume the proof of knowledge is sound, this second case can only happen with negligible probability. Therefore, with overwhelming probability the adversary either submitted a signature previously computed by some group member, or A_0 constructed a new signature using his knowledge of one of the group member's secret key. In the latter case, A_0 already knew what the response of the oracle would be and could have continued the computation without need of the query Q . In the former case, A_0 does acquire knowledge through the interaction, but this knowledge is available to A_1 through its access to the internal state of all group members through time. So with overwhelming probability we can reduce a computation of A_0 to one of A_1 .

Claim (Full anonymity). Under the assumptions of the previous proposition, and assuming further that the signature of knowledge composes well with ElGamal encryption, our group signature scheme provides full anonymity.

Argument. Since the identity of the signer is encrypted using ElGamal, which is semantically secure, it is safe against passive attacks on the encryption scheme, as long as the proofs of knowledge compose well with it. But from the previous

proposition, we know that an adversary does not gain any significant advantage from accessing the OPEN oracle, i.e., from staging active attacks against the encryption scheme.

Remark 2. Such a result may sound surprising, specially in view of the proof in [5], which implies that in order for a group signature scheme to be secure in the formal model it is *required* that the cipher used be secure against chosen ciphertext attacks, whereas our scheme uses ElGamal, which is only semantically secure. Still, in light of results such as [27], it is at least conceivable that semantic security is sufficient if the proofs of knowledge are *non-malleable*.

Moreover, our scheme can be easily modified to use Cramer-Shoup encryption instead of ElGamal. This will only require adding the authenticating tags to each of the two ElGamal encrypted pairs (Y_1, Y_2) and (R_1, R_2) and verifying such tags during signature verification as well as before decrypting within the signature opening algorithm. (Notice that the authenticating tags can be shown well-constructed without requiring knowledge of the Cramer-Shoup scheme's private keys.)

The second property we should prove is the full traceability.

Claim (Full traceability). Under the assumptions of the previous claims, and using the fact that the modified Nyberg-Rueppel signature is unforgeable under chosen message attacks, our group signature scheme is fully traceable.

Argument. To prove such a claim one must show the impossibility of an adversary to produce a signature that, when opened, reveals either an invalid pseudonym or a valid pseudonym whose secret is unknown to the attacker. In each case, the attacker must either be capable of forging the proof of knowledge of a certificate on a pseudonym and associate secret, or must be able to produce certificates for new, invalid users. (Forging a new certificate for a valid, uncompromised user would NOT suffice, for the adversary would still have to prove knowledge of the pseudonym secret.) The latter case is not possible because the modified Nyberg-Rueppel is existentially unforgeable under chosen message attacks. The former case would violate the assumption that the Schnorr signature implements sound proofs-of-knowledge.

C An Alternative Construction in the RSA Ring

In this appendix we briefly describe another possible realization of the scheme. Much of the notation and procedures are the same as in section 4. The shared parameters are chosen differently. We define \mathcal{G} to be the group of quadratic residues in the RSA ring generated by a composite modulus which is a product of safe primes. Namely, a trusted party generates two safe primes p, q , and publishes $n = pq$. After constructing a proof that n is formed correctly, the third party may forget its factorization, as it is not needed for the scheme. The group \mathcal{F} is chosen as a group of order n . For that, one searches for a prime \hat{p} so that $\hat{p} = mn + 1$, where m is a small number. One then sets \mathcal{F} to be the subgroup of m -powers in the group $\mathbf{Z}_{\hat{p}}^*$. The group-specific parameters are the same.

The JOIN protocol is little changed. There are no restrictions on the value of $r = I_U g^{-k} \bmod n$, where k is chosen in the interval $[-2^{\tau+2\kappa}, 2^{\tau+2\kappa} - 1]$; as before, κ stands for the bitlength of $|\mathcal{G}|$. The terms a , b , and s cannot be reduced modulo the unknown order of \mathcal{G} , which is unknown.

Table 5. Shared and group specific parameters.

<u>Shared parameters</u>	
Security parameters $\delta, \epsilon, \sigma_1, \sigma_2, \tau$ (integers);	
Secure hash function $\mathcal{H}(\cdot) : \{0, 1\}^* \rightarrow \{0, 1\}^\tau$;	
n , a composite integer, the product of safe primes;	
\hat{p} , a prime satisfying $\hat{p} = mn + 1$, where m is small;	
$\mathcal{G} = \{x \in \mathbf{Z}_n^* : \exists a \in \mathbf{Z}_n^* \text{ s.t. } x \equiv a^2 \bmod n\}$;	
$\mathcal{F} = \{x \in \mathbf{Z}_{\hat{p}}^* : \exists a \in \mathbf{Z}_{\hat{p}}^* \text{ s.t. } x \equiv a^m \bmod \hat{p}\}$;	
P , an (optional) proof that n is a product of safe primes;	
g, g_1 , and g_2 , generators of \mathcal{G} ;	
P' , an (optional) proof that g, g_1 , and g_2 are quadratic residues.	
<u>Group-specific parameters</u>	
\mathcal{S} , a string including y and y_2 ;	
CA's signature $\text{CERT}_{CA}(\mathcal{S})$.	

Table 6. The JOIN protocol.

$U \rightarrow GM : J_U = I^m \bmod n$
$GM \rightarrow U : a, b \in [-2^{\tau/2+\kappa}, 2^{\tau/2+\kappa} - 1]$
$U \rightarrow GM : \text{Sig}_U(I_U = J_U^a g_1^b \bmod n, PK[u : I_U = g_1^u])$
$GM \rightarrow U : r = I_U g^{-k} \bmod n,$
$s = -xr + k \in [-2^{2\kappa+\tau+1}, 2^{2\kappa+\tau+1} - 1]$

Table 7. The SIGN protocol.

<u>Proof arguments:</u>	
$Y_1, Y_2, R_1, R_2, \chi, E_1.$	
<u>Signature of knowledge:</u>	
$SPK[u, \ell', \ell, r, s, t : Y_1 = g_1^u g_2^{\ell'} \wedge Y_2 = g_2^{\ell'}]$	
$\wedge E_1 = E_1(r, 0) = \chi^r \wedge R_1 = r^{-1} y_2^\ell \wedge R_2 = g_2^\ell$	
$\wedge E_2 = Y_1 R_1 = E_2(r, s, t) = y^r g^s y_2^t \wedge Y_2 R_2 = g_2^t](\mathcal{M})$	

The SIGN protocol can be considerably simplified. There is no need for an extra commitment in a group of unknown order, as the order of the group \mathcal{G} is itself unknown. Moreover, there is no need to prove that the r in the commitment E_1 is bounded in a certain interval, as a cheating U could not find a value that reduces to different values $r_1 \bmod n$ and $r_2 \bmod \phi(n)$ while satisfying the signature equation, because $\phi(n)$ is unknown to U .

Protocol OPEN is unchanged from the previous case.

Accumulating Composites and Improved Group Signing

Gene Tsudik¹ and Shouhuai Xu^{2,*}

¹ Dept. of Information and Computer Science
University of California at Irvine
`gts@ics.uci.edu`

² Department of Computer Science
University of Texas at San Antonio
`shxu@cs.utsa.edu`

Abstract. Constructing practical and provably secure group signature schemes has been a very active research topic in recent years. A group signature can be viewed as a digital signature with certain extra properties. Notably, anyone can verify that a signature is generated by a legitimate group member, while the actual signer can only be identified (and linked) by a designated entity called a group manager. Currently, the most efficient group signature scheme available is due to Camenisch and Lysyanskaya [CL02]. It is obtained by integrating a novel dynamic accumulator with the scheme by Ateniese, et al. [ACJT00].

In this paper, we construct a dynamic accumulator that accumulates *composites*, as opposed to previous accumulators that accumulated *primes*. We also present an efficient method for proving knowledge of factorization of a committed value. Based on these (and other) techniques we design a novel provably secure group signature scheme. It operates in the *common auxiliary string* model and offers two important benefits: 1) the Join process is very efficient: a new member computes only a single exponentiation, and 2) the (unoptimized) cost of generating a group signature is 17 exponentiations which is appreciably less than the state-of-the-art.

1 Introduction

The notion of group signatures was introduced by Chaum and van Heyst in 1991 [CvH91]. Since then, seeking practical and provably secure group signature schemes – and their interactive dual known as identity escrow [KP98] – has been a very active research area in applied cryptography. A group signature can be seen as a normal digital signature with the following extra properties: anyone can verify that a signature is generated by a legitimate group member, while the actual signer can only be identified and linked by a designated entity called a group manager.

The basic idea underlying most group signature schemes (as well as ours) is the following: In order for a group member (Alice) to sign a message, she

* Work done while affiliated with University of California at Irvine.

needs to construct an *authorization-proof* to show that she has a legitimate membership certificate, and an *ownership-proof* to demonstrate knowledge of the secret corresponding to the membership certificate. The issues in these two proofs are similar to those encountered in a normal public key infrastructure (PKI) setting, namely, a signature can be verified using the alleged signer's public key contained in a certificate which has not been revoked. However, the group signature scenario is more complicated, since a signer cannot show her membership certificate without compromising her anonymity. It is precisely this anonymity requirement that makes it very difficult to have a practical solution that facilitates revocation of membership certificates (a concept compatible to certificate revocation in a normal PKI), or the validity check of non-revoked membership certificates.

Early group signature schemes (e.g., [CP94]) have the characteristics that the sizes of the group public key and/or of group signatures *linearly* depend on the number of group members. The advantages of these schemes include: (1) many of the schemes have been proven secure using some standard cryptographic assumptions (such as the hardness of computing discrete logarithms), and (2) *authorization-proof* is trivial since revoking a member is done by the group manager that removes the corresponding membership certificate from the group public key. The disadvantage of such schemes is that the complexity of *ownership-proof*, namely proving and verifying that one knows the secret corresponding to a (non-identified yet non-revoked) membership certificate, is linear in the number of current members and thus becomes inefficient for large groups.

To combat linear complexity incurred as part of *ownership-proof*, Camenisch and Stadler [CS97] took a different approach where the sizes of the group public key and of group signatures are constant and independent of the number of current group members. This approach has been adopted in some follow-on results, e.g., [CM98, CM99a, ACJT00]. As initially presented, these schemes only support adding new members. Since then, [CS97] and [ACJT00] have been extended to support membership revocation [BS01, S01, AST02]. However, revocation incurs certain significant costs due to some (or all) of the following:

- Group manager re-issuing all certificates for each revocation interval.
- Group member (signer) proving, as part of signing, that her certificate is not revoked.
- Verifier checking each group signature against the current list of revoked certificates.

As pointed out in [CL02], each of the above has a linear dependency either on the number of current, or the total number of deleted, members.

State-of-the-Art. Currently, the most efficient group signature scheme is due to Camenisch and Lysyanskaya [CL02]. It is constructed by incorporating a *dynamic accumulator*, which allows efficient *authorization-proofs*, into the group signature scheme due to Ateniese, et al. [ACJT00], which allows efficient *ownership-proofs*. The concept of dynamic accumulators introduced in [CL02] is a variant of the accumulator due to Baric and Pfitzmann [BP97]. It enables a

group member to conduct a light-weight authorization-proof such that both the proving and verifying complexities are independent of the number of the current, or total deleted, members. We note that the use of dynamic accumulators to facilitate *authorization-proofs*, requires the group manager to disseminate certain information, such as the values deleted from the accumulator whenever a member (or a set of thereof) joins or leaves the group.

1.1 Contributions

The main contribution of this paper is a new group signature scheme provably secure against adaptive adversaries, i.e., adversaries allowed to adaptively join and leave the group. The scheme is obtained by integrating several building blocks, some of which are new (e.g., the dynamic composites accumulator), while others are more efficient than previous techniques providing the same functionality (e.g., the multiplication protocol that allows one to prove that she knows the factorization of a committed value). More specifically:

- A new dynamic accumulator that accumulates *composites* (see Section 5.1), as opposed to the prior construct that accumulates *primes* [CL02]. This accumulator fits well into a group signature scheme because it allows us to conduct simultaneous *authorization-proofs* and *ownership-proofs* based on the factorizations of accumulated *composites*.
- A protocol (in Section 5.2) for proving knowledge of factorization of a committed value, which, in our case, corresponds to an accumulated composite. This protocol is more efficient than prior art, such as [DF02].
- A protocol (in Section 5.3) for verifiable encryption of discrete logarithms, based on the public key cryptosystem due to Catalano, et al. [CGHN01]. This protocol is more efficient than previous similar protocols (e.g., the one presented in [MR01]) based on the Paillier cryptosystem [P99].

As mentioned earlier, the state-of-the-art group signature scheme by Camenisch and Lysyanskaya is obtained by integrating a dynamic prime accumulator [CL02] with the *bare* group signature scheme in [ACJT00]. This integration was needed since a prime accumulator cannot be used for *ownership-proof*. In comparison with the [CL02] scheme, our approach has three major benefits:

- Use of the new accumulator construct simultaneously for both *ownership-proof* and *authorization-proof*. This yields a conceptually simpler scheme.
- Efficient Join: a new member only computes a single exponentiation in order to verify that her *composite* has been correctly accumulated. In comparison, Join involves more than 30 exponentiations in [CL02]. We note that this complexity does not stem from the use of the dynamic accumulator; it is inherited from Join of [ACJT00].
- Efficient Sign and Verify: the computational complexity of signing is 17 exponentiations (without any optimizations) which is notably lower than 25 in the Camenisch-Lysyanskaya scheme. A similar gain in efficiency is also achieved in the verification process.

Our scheme also has some potential drawbacks. They are discussed in Section 7.

1.2 Organization

In Section 2, we overview the model and goals of group signatures. Then, in Section 3, we introduce the basic ideas underlying our group signature scheme. Section 4 presents some cryptographic preliminaries and Section 5 describes some building blocks. The new group signature scheme is found in Section 6; its features and potential drawbacks are discussed in Section 7. Due to space limitations, technical details of the security proof and some interesting discussions are deferred to the extended version [TX03].

2 Model and Goals

Participants. A group signature scheme involves a *group manager* (responsible for admitting/deleting members and for revoking anonymity of group signatures, e.g., in cases of dispute or fraud), a set of *group members*, and a set of *signature verifiers*. All participants are modeled as probabilistic polynomial-time interactive Turing machines.

Communication Channels. All communication channels are assumed to be asynchronous. The communication channel between a signer and a receiver is assumed to be anonymous.

Trust. We assume that the *group manager* will not admit unauthorized individuals into the group. This is reasonable, since, otherwise, the group manager can issue valid membership certificates to rogue members and thus make the group signature scheme useless. We assume that the group members, whether honest or not, behave *rationality*. More precisely, a dishonest group member may seek to undermine the system (e.g., by colluding with other internal or external parties) as long as the attack will not be traced back to herself. Nonetheless, she will not take the chance if she (or anyone else colluding with her) is bound to be caught. This assumption is also reasonable since, in any group signature scheme (indeed, in any cryptographic setting), a dishonest user could (for instance) simply give away her own secrets. However, she is bound to be held accountable for any consequences of such misbehavior.

2.1 Definitions

A group signature scheme consists of the following procedures:

- **Setup.** On input a security parameter, this probabilistic algorithm outputs the initial group public key and the secret key for the group manager.
- **Join.** This is a protocol executed between the group manager and a user who is to become a group member. The user's output is a membership certificate and a membership secret; the group manager's output is some updated information that indicates the current state of the system.
- **Revoke.** This is a deterministic algorithm which, on input a membership certificate, outputs some updated information that indicates the current state of the system after revoking the given membership certificate.

- **Update.** This is a deterministic algorithm that may be triggered by any **Join** or **Revoke** operation. It is run by the group members after obtaining certain information from the group manager.
- **Sign.** This is a probabilistic algorithm which, on input of: a group public key, a membership certificate, a membership secret and a message, outputs a group signature.
- **Verify.** This is a deterministic algorithm for establishing the validity of an alleged group signature on a message with respect to the group public key.
- **Open.** This is an algorithm which, on input of: a message, a valid group signature, a group public key and a group manager’s secret key, determines the identity of the actual signer.

2.2 The Goals

A secure group signature scheme must satisfy the following properties:

- **CORRECTNESS.** Any signatures produced by a group member using **Sign** must be accepted by **Verify**.
- **UNFORGEABILITY.** Only group members are able to sign messages on behalf of the group.
- **ANONYMITY.** Given a valid group signature, identifying the actual signer is computationally hard for everyone but the group manager.
- **UNLINKABILITY.** Deciding whether two different group signatures were generated by the same member is computationally hard for everyone but the group manager.
- **NO-FRAMING.** No combination of a group manager and a subset of dishonest group members can sign on behalf of a single honest member. That is, no honest member can be made responsible for a signature she did not produce.
- **TRACEABILITY.** The group manager is always able to identify the actual signer of any valid group signature.
- **COALITION-RESISTANCE.** A colluding subset of group members (even all members) cannot generate a signature that the group manager cannot trace.

3 Basic Ideas

The basic idea underlying our group signature scheme is to utilize an accumulator that accumulates *composites*, where the factorization of a *composite* is only known to the user who generates it. More specifically, suppose a group member has a witness w such that $w^e = v \bmod n$ where v is the public accumulator value and n is the product of two safe primes. The factorization of $e = e_1 e_2$ (i.e., the primes e_1 and e_2) is only known to the member. This knowledge allows the user to conduct an *ownership-proof* by demonstrating that $e = e_1 e_2$. The witness w facilitates an *authorization-proof* that $w^e = v \bmod n$.

While the basic idea is quite simple, we must deal with potential abuses. We now present an informal discussion of some subtleties, and suggest countermeasures. Readers who prefer to commence with the more in-depth technical description may wish to skip this section.

- Q:** How to ensure anonymity while preserving authenticity?
- A:** A signer “encrypts” both w and e such that the required properties regarding them can be shown on the corresponding “ciphertexts”. In particular, a signer needs to show $w^e = v$ for the *authorization-proof*, and $e = e_1 e_2$ for the *ownership-proof*. As long as e is chosen such that it is infeasible to factor, no group of participants (including the group manager) can frame an honest group member.
- Q:** How to deal with multiple dishonest group members who collude (by revealing to each other factorizations of their respective composites) and produce new membership certificates? For example, if Alice chooses $e_1 = e_{1,1} e_{1,2}$ and Bob chooses $e_2 = e_{2,1} e_{2,2}$, they can collude to obtain new membership certificates for the values such as $(e_1 e_2, 1)$ or $(e_{1,1} e_2, 1)$.
- A:** Although we cannot prevent such abuses, we can ensure that, the group manager can factor at least one of the colluding group member’s e (e_1 , or e_2 , or even both) and thus identify at least one of the miscreants. One way to do this, as we shall see, is to use a public key encryption scheme (for which the group manager knows the private key) so that the signer is forced to encrypt an “accumulated” value she is claiming. Note that even a dishonest member cannot afford to encrypt $e_{1,1}$, since, otherwise, the group manager can factor her composite and forge signatures that will be traced back to the dishonest member.
- Q:** How to deal with multiple dishonest group members who collude (but *do not reveal* to each other the factorizations of their composites) and produce new membership certificates? For example, suppose that Alice holds (w_1, e_1) and Bob holds (w_2, e_2) , where $e_1 = e_{1,1} e_{1,2}$, $e_2 = e_{2,1} e_{2,2}$, $w_1^{e_1} = w_2^{e_2} = v$. They can collude and generate $(w', e' = e_1 e_2)$ such that $(w')^{e_1 e_2} = v$.
- A:** We prevent such attacks by requiring all verifiers to check that e' falls within a certain range.
- Q:** Does the group manager need to check whether a composite presented by a new user during Join is well-formed, i.e., a product of two large primes? If not, what if a dishonest group member chooses e to be a single prime or a product of multiple (more than 2) primes?
- A:** We do not aim to prevent such abuses (this also justifies our efficiency gains). However, will be shown, no adversary can gain any benefit from any such abuse since the group manager is always able to identify at least one of the colluding group members. Moreover, choosing appropriate composites is indeed on the user’s behalf.
- Q:** What if the group manager attempts to frame an honest group member by using the group member’s membership certificate (w, e) where $w^e = v$ while providing a proof of factorization of some value $e' \neq e$.
- A:** The Sign process ensures that, if the group manager proves knowledge of the factorization of an “accumulated” value $e' \neq e$, then the witness value that the group manager (or any impersonator) is showing is $w' \neq w$. Moreover, the group manager is required to conduct a zero-knowledge proof as part of Open such that the decryption corresponding to an ElGamal ciphertext (of w) is correct.

4 Preliminaries

Definition 1. (safe RSA modulus). We say $n = pq$ is a safe RSA modulus, if $p = 2p' + 1$, $q = 2q' + 1$, and p , q , p' , q' are all primes.

By convention, let $\gcd(0, n) = n$, and \mathbb{QR}_n be the subgroup of quadratic residues modulo n .

Definition 2. (Strong RSA Problem). Let $n = pq$ be a RSA-like modulus and \mathbb{G} be a cyclic subgroup of \mathbb{Z}_n^* , where $|\text{ord}(\mathbb{G})| = l_{\mathbb{G}}$. Given n and $z \in_R \mathbb{G}$, the Strong RSA Problem consists of finding $w \in \mathbb{G}$ and $e > 1$ such that $z = w^e \bmod n$.

Assumption 1 (Strong RSA Assumption). Suppose a RSA-like modulus n and $z \in_R \mathbb{G}$ are obtained according to a given security parameter $l_{\mathbb{G}}$. The assumption states that any probabilistic polynomial-time algorithm \mathcal{A} can solve the Strong RSA Problem with only negligible probability.

The following lemma is useful and has appeared in many places (e.g., [GKR00]).

Lemma 1. Suppose $n = pq$ is a safe RSA modulus. Given an element $w \in \mathbb{Z}_n^* \setminus \{1, -1\}$ of $\text{ord}(w) < p'q'$, either $\gcd(w - 1, n)$ or $\gcd(w + 1, n)$ is a prime factor of n .

Definition 3. (Decisional Diffie-Hellman Problem). Let $\mathbb{G} = \langle g \rangle$ be a cyclic group generated by g , where $|\text{ord}(\mathbb{G})| = l_{\mathbb{G}}$. Given g , g^x , g^y , and $g^z \in_R \mathbb{G}$, the Decisional Diffie-Hellman Problem consists of deciding whether $g^{xy} = g^z$.

Assumption 2 (Decisional Diffie-Hellman Assumption). Suppose a group \mathbb{G} and an element g of order $\text{ord}(\mathbb{G})$ are obtained according to a given security parameter $l_{\mathbb{G}}$. The assumption states that there is no probabilistic polynomial-time algorithm that distinguishes with non-negligible probability (g, g^x, g^y, g^{xy}) from (g, g^x, g^y, g^z) , where $x, y, z \in_R \mathbb{Z}_{\text{ord}(\mathbb{G})}$.

We will utilize the ElGamal public key cryptosystem [E85] whose semantic security is based on DDHA [TY98]. Since we always work in the setting of modulo a safe RSA modulus, we need certain group in which the DDHA holds.

Fact 1 If n is a safe RSA modulus, then \mathbb{QR}_n is a cyclic subgroup of order $p'q'$. Moreover, if $a \in \mathbb{Z}_n^*$ and $\gcd(a \pm 1, n) = 1$, then $g = a^2 \bmod n$ is of order $p'q'$.

4.1 The CGHN Public Key Cryptosystem

We now briefly review Paillier's cryptosystem [P99]. Suppose $n = pq$ where p and q are large primes. Then we have Euler's Totient function $\phi(n) = (p - 1)(q - 1)$ and Carmichael's function $\lambda(n) = \text{lcm}(p - 1, q - 1)$. It follows that: $w^{\lambda(n)} = 1 \bmod n$ and $w^{n \cdot \lambda(n)} = 1 \bmod n^2$ for any $w \in \mathbb{Z}_{n^2}^*$. Let $(n, g; n, g, p, q)$ be a pair of Paillier public and private keys as specified in [P99]. To encrypt a message $m \in \mathbb{Z}_n$, one chooses $r \in_R \mathbb{Z}_n^*$ and computes the ciphertext $c = g^{mr^n} \bmod n^2$. Note that an interesting selection of g is $g = (1 + n)$ because $(1 + n)^m = 1 + mn \bmod n^2$.

A performance disadvantage of the Paillier cryptosystem is that one needs to compute $r^n \bmod n^2$. Catalano et al. [CGHN01] observed that if we always set $g = (1 + n)$ then we can use any public exponent t as long as $\gcd(t, \lambda(n^2)) = 1$, because a ciphertext $c = (1 + mn)r^t \bmod n^2$ yields $c = r^t \bmod n$, thereby r can be recovered by a standard RSA decryption operation. This means that one only needs to compute an exponentiation operation modulo n^2 with respect to an exponent $|t| \ll |n|$. We call this variant the CGHN cryptosystem whose semantic security is based on the following DSRA assumption.

Definition 4. (Computational Small t -roots Problem). *This is a variant of the RSA problem in $\mathbb{Z}_{n^2}^*$. The problem is to invert $y^t \bmod n^2$, where $y \in \mathbb{Z}_n$, $t \in \mathbb{Z}_n$, and $\gcd(t, \lambda(n^2)) = 1$.*

Definition 5. (Decisional Small Residuosity Problem, DSRP). *This is a decisional version of the above computational problem. Given an element $x \in_R \mathbb{Z}_{n^2}^*$, one needs to decide whether x is the form y^t with $y \in \mathbb{Z}_n$.*

Assumption 3 (Decisional Small Residuosity Assumption, DSRA) *Let n be a randomly chosen l -bit RSA modulus, $t \in \mathbb{Z}_n$ such that $\gcd(t, \lambda(n^2)) = 1$, and $x \in_R \mathbb{Z}_{n^2}^*$. There exists no probabilistic polynomial-time algorithm that is able to decide, with non-negligible advantage, whether x is the form y^t with $y \in \mathbb{Z}_n$.*

The following lemma will be used (the proof is deferred to [TX03]).

Lemma 2. *Suppose n is a safe RSA modulus. If $A^a = 1 \bmod n^2$ where $A \in \mathbb{Z}_{n^2}^*$ and $\gcd(a, n \cdot \lambda(n)) = 1$ or 2, then $A = \pm 1 \bmod n^2$.*

5 Building Blocks

5.1 A Composite Accumulator

Definition 6. *A dynamic accumulator for a family of inputs $\{\mathfrak{X}_l\}$ is a family of families of functions $\{\mathcal{F}_l\}$ with the following properties:*

- **GENERATION.** *There is an efficient probabilistic algorithm \mathcal{G} that on input 1^l produces a random element f of \mathcal{F}_l , and some auxiliary information aux_f about f .*
- **EVALUATION.** *$f \in \mathcal{F}_l$ is a polynomial-size circuit that, on input $(u, x) \in \mathfrak{U}_f \times \mathfrak{X}_l$, outputs a value $v \in \mathfrak{U}_f$, where \mathfrak{U}_f is an efficiently-samplable input domain for the function f , \mathfrak{X}_l is the intended input domain whose elements (i.e., composites) are to be accumulated.*
- **QUASI-COMMUTATIVE.** *For all l , for all $f \in \mathcal{F}_l$, for all $u \in \mathfrak{U}_f$, for all $x_1, x_2 \in \mathfrak{X}_l$, $f(f(u, x_1), x_2) = f(f(u, x_2), x_1)$. If $\mathfrak{X} = \{x_1, \dots, x_m\} \subset \mathfrak{X}_l$, then by $f(u, \mathfrak{X})$ we denote $f(\dots f(f(u, x_1), \dots), x_m)$.*
- **WITNESS.** *Let $v \in \mathfrak{U}_f$ and $x \in \mathfrak{X}_l$. A value $w \in \mathfrak{U}_f$ is called a witness for x in v under f if $v = f(w, x)$.*
- **ADDITION.** *Let $f \in \mathcal{F}_l$, and $v = f(u, \mathfrak{X})$ be the accumulator so far. There is an efficient algorithm \mathcal{A} to accumulate a given value $x' \in \mathfrak{X}_l$. The algorithm outputs: (1) $\mathfrak{X}' = \mathfrak{X} \cup \{x'\}$ and $v' = f(v, x') = f(u, \mathfrak{X}')$; (2) w' which is the witness for $x \in \mathfrak{X}$ in v' .*

- **DELETION.** Let $f \in \mathcal{F}_l$, and $v = f(u, \mathfrak{X})$ be the accumulator so far. There exist efficient algorithms \mathcal{D} , \mathcal{W} to delete an accumulated value $x' \in \mathfrak{X}$. The functionality of the algorithms includes: (1) $\mathcal{D}(\text{aux}_f, v, x') = v'$ such that $v' = f(u, \mathfrak{X} \setminus \{x'\})$, and (2) $\mathcal{W}(w, x, x', v, v') = w'$ such that $f(w', x) = v'$, where $x \in \mathfrak{X}$ and $f(w, x) = v$.

Definition 7. Let $\mathcal{U}'_f \times \mathfrak{X}'_l$ denote the domains for which the function $f \in \mathcal{F}_l$ is defined (thus $\mathcal{U}_f \subseteq \mathcal{U}'_f$, $\mathfrak{X}_l \subseteq \mathfrak{X}'_l$). To capture security of a dynamic accumulator accumulating composites, we consider the following game: At the beginning of the game, an accumulator manager sets up the function f and the value u and hides the trapdoor information aux_f . Then, the adversary \mathcal{ADV} is allowed to adaptively modifies the set, \mathfrak{X} , of accumulated values: When a value $x \in \mathfrak{X}_l$ is added, the manager updates the accumulator value using algorithm \mathcal{A} ; when a value $x \in \mathfrak{X}$ is deleted, the manager algorithm \mathcal{D} publishes the result. We say \mathcal{ADV} wins in this game, if it, with non-negligible probability, manages to output a witness w' for a value $x' \in \mathfrak{X}'_l$ such that $x' \nmid \prod_{x \in \mathfrak{X}} x$. More formally, we require that:

$$\Pr[(f, \text{aux}_f) \leftarrow \mathcal{G}(1^l); u \leftarrow \mathcal{U}_f; (w, x', \mathfrak{X}) \leftarrow \mathcal{ADV}^{\mathcal{O}_{add}, \mathcal{O}_{del}}(f, u, \mathcal{U}_f) : \\ w' \in \mathcal{U}'_f; x' \in \mathfrak{X}'_l; x' \nmid \prod_{x \in \mathfrak{X}} x; f(w', x') = f(u, \mathfrak{X})]$$

to be negligible, where \mathcal{O}_{add} (\mathcal{O}_{del}) is the oracle for the ADDITION (resp. DELETION) operations. (Note that only a legitimately accumulated value x must belong to \mathfrak{X}_l , whereas a forged value x' can belong to a possibly larger set \mathfrak{X}'_l .)

Construction. This construction is a variant of the one in [CL02].

- \mathcal{F}_l is the family of functions that correspond to exponentiation modulo safe RSA modulus drawn from the integers of length l . Choosing $f \in \mathcal{F}_l$ amounts to choosing a random safe RSA modulus $n = pq$ of length l , where $p = 2p' + 1$, $q = 2q' + 1$. We will denote by f the function corresponding to modulus n and domain $\mathfrak{X}_{A,B}$ by $f_{n,A,B}$.
- $\mathfrak{X}_{A,B} = \{e_1 e_2 : e_1 \in \mathfrak{S}_1 \wedge e_2 \in \mathfrak{S}_2\}$, where $\mathfrak{S}_1 = \{e : e \in \text{primes} \wedge e \neq p' \wedge e \neq q' \wedge A_1 \leq e \leq B_1\}$, $\mathfrak{S}_2 = \{e : e \in \text{primes} \wedge e \neq p' \wedge e \neq q' \wedge A_2 \leq e \leq B_2\}$, A_1 , A_2 , B_1 , and B_2 can be chosen with arbitrary polynomial dependence on the security parameter l as long as $4 < A_1$, $1 < A_2$, $B_1 < A_1^2$, $B_2 < A_1^2$, and $B_1 B_2 < p' q'$. Then, $\mathfrak{X}'_{A,B} \subseteq \{5, \dots, A_1^4 - 1\}$ and $\mathfrak{X}_{A,B} \subseteq \mathfrak{X}'_{A,B}$.
- For $f = f_{n,A,B}$, the auxiliary information aux_f is the factorization of n .
- For $f = f_{n,A,B}$, $\mathcal{U}_f = \{u \in \mathbb{Q}\mathbb{R}_n : u \neq 1\}$ and $\mathcal{U}'_f = \mathbb{Z}_n^*$.
- For $f = f_{n,A,B}$, $f(w, x) = w^x \bmod n$. We remark that $f(f(w, x_1), x_2) = f(w, \{x_1, x_2\}) = w^{x_1 x_2} \bmod n$.
- Update of the accumulator value. Adding a value x' to the accumulator value v is done by setting $v' = f(v, x') = v^{x'} \bmod n$. Deleting a value x' from the accumulator is done by setting $v' = \mathcal{D}((p, q), v, x') = v^{(x')^{-1} \bmod \phi(n)} \bmod n$.
- Update of witness. Updating the witness w after x' has been added can be done by $w' = f(w, x') = w^{x'}$. In the case that $x' \neq x \in \mathfrak{X}_{AB}$ has been deleted from the accumulator, the witness w can be updated as follows. By the extended GCD algorithm, one can compute $\alpha, \beta \in \mathbb{Z}$ such that $\alpha x + \beta x' = 1$

and then $w' = \mathcal{W}(w, x, x', v, v') = (v')^\alpha w^\beta$. This guarantees $f(w', x) = (w')^x = v' \bmod n$ because:

$$\begin{aligned} w' &= (v')^\alpha w^\beta = (v^{(x')^{-1} \bmod \phi(n)})^\alpha w^\beta = w^{(\alpha x + \beta x')((x')^{-1} \bmod \phi(n))} \\ &= w^{(x')^{-1} \bmod \phi(n)} \bmod n. \end{aligned}$$

Note that it is crucial $(x', \phi(n)) = 1$, but this is always guaranteed.

Theorem 1. ([TX03]) *Under the Strong RSA Assumption (SRSA), the above construction is a secure dynamic accumulator that accumulates composites.*

5.2 Proving That One Knows the Factorization of a Committed Value

In order to enable *ownership-proofs*, we adopt the Damgard-Fujisaki commitment scheme [DF02] with slight modification. Nonetheless, our protocol for a signer to prove that she knows the factorization of a committed value is more efficient than the protocol presented in [DF02], and thus may be independently interesting.

The Commitment Scheme. Let l (for the length of the modulus) and k (for challenge length) be security parameters, where $l \gg k$. This scheme consists of the following three algorithms.

- **Set-up.** This algorithm is run by a trusted third party (TTP). Given a security parameter l , TTP chooses a safe RSA modulus $N = PQ$, where $P = 2P' + 1$, $Q = 2Q' + 1$, and $|P'| = |Q'| = l/2$. Denote by $\mathbb{G} = \mathbb{QR}_N$ and $l_{\mathbb{G}} = |\text{ord}(\mathbb{G})| = l$. TTP chooses two generators of \mathbb{G} , G and H , uniformly at random; i.e., $\mathbb{G} = \langle G \rangle = \langle H \rangle$. Note that Fact 1 implies that this can be easily done.
- **Commit.** To commit to an integer x , the prover chooses $r \in_R \mathbb{Z}_{[N/4]}$ and sends $C = H^x G^r \bmod N$ to the verifier.
- **Open.** To open a commitment, the prover must send x, r, b such that $C = H^x G^r b \bmod N$, $b = \pm 1$.

Lemma 3. ([DF02]) *The above commitment scheme is perfectly hiding and computationally binding.*

A Protocol for Proving That One Knows the Factorization of a Committed Value. Suppose X is a given random integer such that $|X| = \lambda_1$. Let $\epsilon > 1$ be a security parameter for statistical zero-knowledge, λ_2 denote length such that $l/2 > \lambda_1 > \epsilon(\lambda_2 + k) + 2$. Alice who holds e is to prove that she knows the factorization of $e = e_1 e_2$, where $e_1 \in \{X - 2^{\lambda_2}, \dots, X + 2^{\lambda_2}\}$ and $e_2 \neq 0, \pm 1$. The protocol goes as follows.

1. The prover, Alice, chooses $r_1 \in_R \pm\{0, 1\}^{l+k}$ and generates $C_1 = H^{e_1} G^{r_1} \bmod N$, $C_3 = (C_1)^{e_2} \bmod N$. In order to prove the knowledge of $e = e_1 e_2$, $e_1, e_2, r_1, r = r_1 e_2$ such that

$$C_1 = H^{e_1} G^{r_1} \bmod N \wedge C_3 = H^e G^r \bmod N \wedge C_3 = (C_1)^{e_2} \bmod N,$$

she executes as follows:

- choose $e'_1 \in_R \pm\{0, 1\}^{\epsilon(\lambda_2+k)}$, $e'_2 \in_R \pm\{0, 1\}^{\epsilon(\lambda_1+k+1)}$, $e' \in_R \pm\{0, 1\}^{\epsilon(2\lambda_1+k+1)}$, $r'_1 \in_R \pm\{0, 1\}^{\epsilon(l+2k)}$, $r' \in_R \pm\{0, 1\}^{\epsilon(l+\lambda_2+2k+1)}$.
 - compute $C'_1 = H^{e'_1} G^{r'_1} \bmod N$, $C'_{3a} = H^{e'} G^{r'} \bmod N$, $C'_{3b} = (C_1)^{e'_2} \bmod N$.
 - send $(C_1, C_3, C'_1, C'_{3a}, C'_{3b})$ to the verifier.
2. The verifier, Bob, chooses $c \in_R \{0, 1\}^k$ and sends c to Alice.
 3. Alice sends Bob $(s_{e_1}, s_{r_1}, s_{e_2}, s_e, s_r)$, where $s_{e_1} = e'_1 - c(e_1 - X)$, $s_{r_1} = r'_1 - c \cdot r_1$, $s_e = e' - c \cdot e$, $s_r = r' - cr$, $s_{e_2} = e'_2 - c \cdot e_2$ (all in \mathbb{Z}).
 4. Bob accepts if the following holds: $H^{s_{e_1}} G^{s_{r_1}} = C'_1 C_1^{-c} H^{e_2 \lambda_1} \bmod N$, $H^{s_e} G^{s_r} = C'_{3a} C_3^{-c} \bmod N$, $C_1^{s_{e_2}} = C'_{3b} C_3^{-c} \bmod N$, $s_{e_1} \in \{-2^{\epsilon(\lambda_2+k)+1}, \dots, 2^{\epsilon(\lambda_2+k)+1}\}$, $s_{e_2} \in \{-2^{\epsilon(\lambda_1+k+1)+1}, \dots, 2^{\epsilon(\lambda_1+k+1)+1}\}$, $s_e \in \{-2^{\epsilon(2\lambda_1+k+1)+1}, \dots, 2^{\epsilon(2\lambda_1+k+1)+1}\}$, $C_3 \neq 1$, and $C_3 \neq (C_1)^b \bmod N$ where $b = \pm 1$.

The proof of the following lemma is available in [TX03].

Lemma 4. *The above protocol is an honest verifier statistical zero-knowledge proof of knowledge e, e_1, e_2 such that $e = e_1 e_2$, $e_1 \in \{X - 2^{\epsilon(\lambda_2+k)+2}, \dots, X + 2^{\epsilon(\lambda_2+k)+2}\}$, $e_2 \in \{-2^{\epsilon(\lambda_1+k+1)+2}, \dots, 2^{\epsilon(\lambda_1+k+1)+2}\} \setminus \{0, \pm 1\}$, $e \in \{-2^{\epsilon(2\lambda_1+k+1)+2}, \dots, 2^{\epsilon(2\lambda_1+k+1)+2}\}$.*

5.3 Verifiable Encryption of a Committed Value

In order to facilitate the **Open** process, we need to force the signer to present an encryption of her accumulated value e for which she proves that she knows its non-trivial factorization $e = e_1 e_2$. For this purpose, we need a verifiable encryption scheme. Here we present such a scheme based on the CGHN public key cryptosystem.

Specifically, suppose public values N , G , and H are chosen according to the commitment scheme in Section 5.2. Let $pk = \langle n, t \rangle$ be a CGHN public key and $sk = \langle n, t, p, q \rangle$ be the corresponding private key, where $n = pq$, $|n| = |N|$, and t is a prime such that $|t| > k$. The prover generates a ciphertext $Y = (1+n)^x r^t \bmod n^2$ and a commitment $C = H^x G^z \bmod N$, where $r \in \mathbb{Z}_n^*$ and $z \in_R \mathbb{Z}_{\lfloor N/4 \rfloor}$. The prover needs to show that the ciphertext Y indeed corresponds to the committed secret x . The protocol is as follows:

1. The prover chooses $x' \in_R \pm\{0, 1\}^{\epsilon(l_2+k)}$, $r' \in_R \mathbb{Z}_n^*$, $z' \in_R \{0, 1\}^{\epsilon(l+k)}$, computes and sends to the verifier $Y' = (1+n)^{x'} (r')^t \bmod n^2$ and $C' = H^{x'} G^{z'} \bmod N$.
2. The verifier responses with a random challenge $c \in_R \{0, 1\}^k$.
3. The prover sends to the verifier $s_x = x' - cx$ (in \mathbb{Z}), $s_r = r^{-c} r' \bmod n^2$, and $s_z = z' - cz$ (in \mathbb{Z}).
4. The verifier accepts if the following holds: $s_x \in \{-2^{\epsilon(l_2+k)+1}, 2^{\epsilon(l_2+k)+1}\}$, $(1+n)^{s_x} (s_r)^t = Y' Y^{-c} \bmod n^2$, and $H^{s_x} G^{s_z} = C' C^{-c} \bmod N$.

Lemma 5. ([TX03]) *The above protocol is an honest-verifier statistical zero-knowledge proof of knowledge x, r, z .*

6 A New Group Signature Scheme

As highlighted in Section 3, the basic idea underlying our group signature scheme is to utilize an accumulator accumulating *composites* such as $e = e_1 e_2$, where e_1 and e_2 are only known to the user who generates it. Suppose v is the accumulator value. This knowledge allows the user to conduct an *ownership-proof* by demonstrating that she knows the factorization of a committed e , whereas the witness w facilitates an *authorization-proof* that $w^e = v \bmod n$.

6.1 Setup

Initialization of the system includes that a group manager establishes some cryptographic parameters and that a TTP establishes some common auxiliary strings. Specifically:

1. Let l , k , and $\epsilon > 1$ be security parameters. Let X be a random integer of length $|X| = \lambda_1$. Suppose λ_2 denotes length such that $l/2 > \lambda_1 > \epsilon(\lambda_2 + k) + 2$. Denote by $A = X - 2^{\lambda_2}$ and $B = X + 2^{\lambda_2}$. Define the integral ranges that $A_1 = \{A, \dots, B\}$, $A_2 = \{2^{\lambda_1}, \dots, 2^{\lambda_1+1} - 1\}$, and $\Gamma = \{-2^{2\lambda_1+1}, \dots, 2^{2\lambda_1+1}\}$. Define $\mathfrak{X}_{A,B} = \{e_1 e_2 : e_1 \in \mathfrak{S}_1 \wedge e_2 \in \mathfrak{S}_2\}$, where $\mathfrak{S}_1 = \{e : e \in \text{primes} \wedge e \in A_1\}$ and $\mathfrak{S}_2 = \{e : e \in \text{primes} \wedge e \in A_2\}$. We assume that no probabilistic polynomial-time (in l) algorithm is able to factor $e \in_R \mathfrak{X}_{A,B}$; this is where we need the stronger factoring assumption (see Section 7 for more discussion). Note that we have (1) $4 < A$, (2) $B(2^{\lambda_1+1} - 1) < A^3$. Let $\mathfrak{X}'_{A,B} \subseteq \{5, \dots, A^3 - 1\}$ such that $\mathfrak{X}_{A,B} \subseteq \mathfrak{X}'_{A,B}$. The group manager executes as follows:
 - It chooses a safe RSA modulus $n = (2p' + 1)(2q' + 1)$ such that $|p'| = |q'| = l/2$. This uniquely determines \mathbb{QR}_n , the quadratic residues subgroup modulo n .
 - It establishes an instance of ElGamal public key cryptosystem. Let $\langle y_1 = g_1^{x_1} \bmod n; x_1 \rangle$ be the pair of public and private keys such that $g_1 \in_R \mathbb{QR}_n$ and $x_1 \in_R \mathbb{Z}_{p'q'}^*$.
 - It establishes an instance of CGHN cryptosystem. Let $\langle n, t; n, t, p, q \rangle$ be the pair of public and private keys, where t is a prime such that $|t| > k$.
 - It establishes an instance of the dynamic accumulator by choosing $u \in_R \mathbb{QR}_n$, establishing (currently empty) public archives \mathfrak{A} for storing values corresponding to added group members, and \mathfrak{D} for storing values corresponding to deleted group members.

The public and private parameters of the group manager are $(n, t, g_1, y_1, u, \mathfrak{A}, \mathfrak{D}, \mathfrak{X}_{A,B}, \mathfrak{X}'_{A,B})$ and (p', q') , respectively. Note that a signature receiver can verify group signatures without knowing the dynamically updated \mathfrak{A} or \mathfrak{D} .

2. Given a security parameter l , a TTP initializes a safe RSA modulus $N = (2P' + 1)(2Q' + 1)$, where $|P'| = |Q'| = l/2$. It also chooses and publishes two random elements $G, H \in_R \mathbb{QR}_N$, where the logarithm of G and H to each other is unknown to any participant in the group signature scheme.

6.2 Join

This protocol is executed between a group member, Alice, and the group manager.

1. Alice chooses two primes $e_1 \in_R \mathfrak{S}_1$ and $e_2 \in_R \mathfrak{S}_2$. This step can be done before the execution of the protocol.
2. Alice sends $e = e_1 e_2$ (in \mathbb{Z}) to the group manager.
3. If $A \cdot 2^{\lambda_1} < e < B \cdot (2^{\lambda_1+1} - 1)$, e is odd, and $e \notin \mathfrak{A}$, the group manager stores Alice's membership certificate (v, e) where v is the current accumulator value (when the first user joins the group, $v = u$). It also updates v in the public key file as $v' = f_n(v, e)$, and adds e to \mathfrak{A} .
4. Alice gets her membership certificate (w, e) and checks if $f_n(w, e) = w^e = v' \bmod n$, where $w = v$.

Remark. The Join process is very efficient (1 exponentiation for both group manager and new user) because of the following: If a dishonest user, Eve, does not choose e that is hard to factor, then any participant (internal or external) who can find certain non-trivial factor of e may be able to sign on her behalf.

6.3 Revoke

Suppose Eve, who has membership certificate (w, e) , is to be expelled from the group. Then the group manager can revoke her membership by updating the current accumulator value v in the public key file: It simply sets $v' = \mathcal{D}(\phi(n), v, e)$, deletes e from \mathfrak{A} , and adds e to \mathfrak{D} .

6.4 Update

Whenever there is a Join and/or Revoke event, the group manager updates the accumulator value from v to v' . Correspondingly, every group member needs to update her membership certificate. An entry in the archives is called “new” if it was entered after the last time a legitimate group member performed an update. Suppose Bob holds a membership certificate (w, e) such that $f_n(w, e) = v$. Then, he updates his membership certificate to (w', e) such that $f_n(w', e) = v'$:

- For all new $e^* \in \mathfrak{A}$, $w'' = f_n(w, \prod e^*)$ and $v'' = f_n(v, \prod e^*)$.
- For all new $e^* \in \mathfrak{D}$, $w' = \mathcal{W}(w'', e, \prod e^*, v'', v')$.

6.5 Sign

Recall that $\langle n, t \rangle$ is the group manager's CGHN public key, and that $y_1 = g_1^{x_1} \bmod n$ is the group manager's ElGamal public key. Suppose that v is the current accumulator value, and that Alice holds (w, e) such that $w^e = v \bmod n$, where $e = e_1 e_2$. Given a message m , Alice generates a group signature as follows.

1. She executes as follows.
 - She chooses $r_1 \in_R \mathbb{Z}_n^*$ and computes a CGHN ciphertext $\delta = (1 + en)r_1^t \bmod n^2$.

- She chooses $r_2 \in_R \pm\{0, 1\}^{l+k}$ and computes an ElGamal ciphertext (α, β) where $\alpha = g_1^{r_2} \bmod n$ and $\beta = w \cdot y_1^{r_2} \bmod n$.
- She chooses $r_4 \in_R \pm\{0, 1\}^{l+k}$ and generates commitments $\sigma = H^{e_1} G^{r_4} \bmod N$, $\tau = \sigma^{e_2} = H^e G^{r_4 e_2} \bmod N$.

2. She needs to prove the knowledge of:

- (w, e) such that $w^e = v \bmod n$, where w corresponds to the ElGamal ciphertext (α, β) , and e corresponds to the CGHN ciphertext δ .
- e_1 and e_2 such that $e_1 \in \Lambda_1$, $e_2 \in \Lambda_2$, and $e = e_1 e_2 \in \Gamma$.

For this purpose, she needs to prove the knowledge of $e, e_1, e_2, r_1, r_2, r_3 = r_2 e, r_4, r_5 = r_4 e_2$ such that:

$$\begin{aligned} \delta &= (1+n)^{e r_1^t} \bmod n^2 \bigwedge \\ \alpha &= g_1^{r_2} \bmod n \bigwedge v = \beta^e \left(\frac{1}{y_1}\right)^{r_3} \bmod n \bigwedge 1 = \alpha^e \left(\frac{1}{g_1}\right)^{r_3} \bmod n \bigwedge \\ \tau &= H^e G^{r_5} \bmod N \bigwedge \sigma = H^{e_1} G^{r_4} \bmod N \bigwedge \tau = \sigma^{e_2} \bmod N \bigwedge \\ e &\in \Gamma \bigwedge e_1 \in \Lambda_1 \bigwedge e_2 \in \Lambda_2. \end{aligned}$$

Specifically, she executes as follows:

(a) She executes the following steps:

- Choose $e' \in \pm\{0, 1\}^{\epsilon(2\lambda_1+k+1)}$ and $r'_1 \in_R \mathbb{Z}_n^*$, and compute $\delta' = (1+n)^{e'} (r'_1)^t \bmod n^2$.
- Choose $r'_2 \in_R \pm\{0, 1\}^{\epsilon(l+2k)}$, $r'_3 \in_R \pm\{0, 1\}^{\epsilon(l+2\lambda_1+2k+1)}$, and generate:

$$\alpha' = g_1^{r'_2} \bmod n, \quad v' = \beta^{e'} \left(\frac{1}{y_1}\right)^{r'_3} \bmod n, \quad \omega' = \alpha^{e'} \left(\frac{1}{g_1}\right)^{r'_3}.$$

- Choose $e'_1 \in_R \pm\{0, 1\}^{\epsilon(\lambda_2+k)}$, $e'_2 \in \pm\{0, 1\}^{\epsilon(\lambda_1+k+1)}$, $r'_4 \in_R \pm\{0, 1\}^{\epsilon(l+2k)}$, $r'_5 \in_R \pm\{0, 1\}^{\epsilon(l+\lambda_1+2k+1)}$, and generate:

$$\tau'_1 = H^{e'} G^{r'_5} \bmod N, \quad \sigma' = H^{e'_1} G^{r'_4} \bmod N, \quad \tau'_2 = \sigma^{e'_2} \bmod N.$$

- (b) She computes $c = \mathcal{H}(m, n, t, g_1, y_1, N, G, H, \delta, \alpha, \beta, \tau, \sigma, \delta', \alpha', v', \omega', \tau'_1, \sigma', \tau'_2)$, where $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^k$ behaves like a random oracle.
- (c) She computes (all the operations, except the computation of s_{r_1} , are in \mathbb{Z}):

$$\begin{aligned} s_e &= e' - c \cdot e, & s_{e_1} &= e'_1 - c \cdot (e_1 - X), & s_{e_2} &= e'_2 - c \cdot e_2, \\ s_{r_1} &= r_1^{-c} \cdot r'_1 \bmod n^2, & s_{r_2} &= r'_2 - c \cdot r_2, & s_{r_3} &= r'_3 - c \cdot r_3, \\ s_{r_4} &= r'_4 - c \cdot r_4, & s_{r_5} &= r'_5 - c \cdot r_5. \end{aligned}$$

- (d) She sends Bob $(m, c, n, t, g_1, y_1, N, G, H, \delta, \alpha, \beta, \sigma, \tau, s_e, s_{e_1}, s_{e_2}, s_{r_1}, s_{r_2}, s_{r_3}, s_{r_4}, s_{r_5})$.

Cost: Our Sign requires 17 exponentiations, whereas [CL02] requires 25 exponentiations. Note that 2 of our 17 exponentiations are $r^t \bmod n^2$ but $t \ll n$ (e.g., $|t| = 161$). See [TX03] for further discussions.

6.6 Verify

Given $(m, c, n, t, g_1, y_1, N, G, H, \delta, \alpha, \beta, \sigma, \tau, s_e, s_{e_1}, s_{e_2}, s_{r_1}, s_{r_2}, s_{r_3}, s_{r_4}, s_{r_5})$, Bob checks if it is a valid signature as follows.

1. Bob computes $c' = \mathcal{H}(m, n, t, g_1, y_1, N, G, H, \delta, \alpha, \beta, \tau, \sigma, \delta', \alpha', v', \omega', \tau'_1, \sigma', \tau'_2)$, where

$$\begin{aligned} \delta' &= (1+n)^{s_e} (s_{r_1})^t \delta^c \bmod n^2, & \alpha' &= g_1^{s_{r_2}} \alpha^c \bmod n, \\ v' &= \beta^{s_e} \left(\frac{1}{y_1}\right)^{s_{r_3}} v^c \bmod n, & \omega' &= \alpha^{s_e} \left(\frac{1}{g_1}\right)^{s_{r_3}} \bmod n, \\ \tau'_1 &= H^{s_e} G^{s_{r_5}} \tau^c \bmod N, & \sigma' &= H^{s_{e_1}-c \cdot 2^{\lambda_1}} G^{s_{r_4}} \sigma^c \bmod N, \\ \tau'_2 &= \sigma^{s_{e_2}} \tau^c \bmod N. \end{aligned}$$

2. Bob accepts if $c = c'$, $s_{e_1} \in \{-2^{\epsilon(\lambda_2+k)+1}, \dots, 2^{\epsilon(\lambda_2+k)+1}\}$, $s_{e_2} \in \{-2^{\epsilon(\lambda_1+k+1)+1}, \dots, 2^{\epsilon(\lambda_1+k+1)+1}\}$, $s_e \in \{-2^{\epsilon(2\lambda_1+k+1)+1}, \dots, 2^{\epsilon(2\lambda_1+k+1)+1}\}$, $\tau \neq 1 \bmod N$, and $\tau \neq \sigma^b \bmod N$ where $b = \pm 1$.

Cost: Verify, without any optimizations, requires 16 exponentiations which is somewhat more efficient than 21 exponentiations in [CL02]. However, we believe that the Verify process in the latter is incomplete; a complete version would require a few more exponentiations. See [TX03] for further discussions.

6.7 Open

Given a valid group signature $(m, c, n, t, g_1, y_1, N, G, H, \delta, \alpha, \beta, \sigma, \eta, \tau, s_e, s_{e_1}, s_{e_2}, s_{r_1}, s_{r_2}, s_{r_3}, s_{r_4}, s_{r_5})$, the group manager can identify the signer by decrypting both w and e such that $w^e = v \bmod n$. It also needs to prove that the decryption of w is correct; namely $DLOG(g_1, y_1) = DLOG(\alpha, \beta/w)$.

1. It decrypts the CGHN ciphertext δ to obtain e , and decrypts the ElGamal ciphertext $\langle \alpha, \beta \rangle$ to obtain w . It must hold that $A^3 > e > 1$.
2. There are further two cases.
 - (a) If $e \in \mathfrak{A}$, then it publishes: (1) the values w and e , and (2) the proof that $DLOG(g_1, y_1) = DLOG(\alpha, \beta/w)$. Note that knowing w and e does not expose neither previous, nor future (even if the system policy allows), signatures generated by the same group member.
 - (b) If $e \notin \mathfrak{A}$, then it must hold that $e \mid \prod_{v \in \mathfrak{A}} v$. Therefore, there must exist $e' \in \mathfrak{A}$ such that $e' > \gcd(e, e') > 1$. Therefore, the group member corresponding to accumulated e' is identified (and revoked).

6.8 Analysis

Theorem 2. ([TX03]) *The above scheme is a secure group signature scheme.*

Corollary 1. *The interactive version of the above group signature scheme is a secure identity escrow scheme.*

7 Discussion

On Factorization Assumption. For typical group signature applications we suggest that the group manager use 2048-bit RSA moduli. For other parameters, we suggest (as an example): $\lambda_1 = 950$, $\lambda_2 = 700$, $\epsilon = 1.1$, $k = 160$. This means that we assume the hardness of factoring large 2-prime composites, where $(\lambda_1 - \lambda_2)$ high-order bits of one prime are known. This assumption is stronger than the standard factorization assumption. However, despite the fixed prefix, it still seems reasonable to assume the hardness of factoring such a composite. (We note that a very similar assumption was used before, e.g., by Camenisch and Michels in [CM98].) Given partial knowledge of the factorization, the best factoring algorithm currently available indicates that, if the higher 475-bits of a prime factor are known, then one can factor n [C96]. Beyond that, no better result is available [C03]. Note that if the higher bits of one prime factor are known, then the higher bits of another factor are also exposed. Nevertheless, knowing $\langle \sigma, \tau = \sigma^{e_2} \bmod N \rangle$ still requires an adversary to compute e_2 in $O(2^{350})$ time (see [G00] and the references therein).

“Lazy” Accumulator Update? In a group signature scheme based on a dynamic accumulator, it is necessary for both signer and verifier to get the updated accumulator whenever there is a member leaves. In the Camenisch-Lysyanskaya scheme, they suggest a nice trick whereby a Join may not have to trigger a group member to get the updated accumulator value. While this trick enables potential gain in communications, it may incur some serious problems in practice. Consider the following scenario: since Alice is lazy, she does not contact the group manager to check the current accumulator value. Instead, she waits for a broadcast message from the group manager. If this message is blocked by an adversary, there is no way for Alice to tell if there has been an accumulator update. Consequently, Alice would generate a group signature which is valid with respect to the outdated accumulator value, i.e., the previous accumulator incarnation. However, the signature is invalid with respect to the current accumulator value. It is unclear how a potential dispute involving this signature can be resolved. At best, the verifier can abuse such a signature.

We suggest that Alice should be diligent and prevent such anomalies by actively querying the group manager for the current accumulator value. This way, if she does not elicit any reply from the group manager, she can simply refuse to generate any group signatures.

On TTP Presence. Our scheme operates in the *common auxiliary string* model which assumes a common string (the specification of a commitment scheme) generated by a trusted third party (TTP) and made available to all participants. The inconvenience posed by this is not significant owing to the following mitigating factors:

- The TTP’s role is only to initialize the cryptographic setting of a commitment scheme. In fact, the TTP can simply disappear after publishing the commitment scheme parameters since it is not involved in any future transactions.

- A single TTP could serve multiple group signature settings, thereby amortizing the complexity. Moreover, threshold cryptography can be used to implement a distributed TTP (see [ACS02]).
- Currently, the most efficient method of obtaining identity escrow schemes (such as [KP98]) that are *concurrently secure* is based on the existence of common auxiliary strings [D00]. Therefore, the identity escrow scheme derived from our group signature scheme can be made concurrently secure without incurring any extra complexity.

8 Conclusion

We presented a dynamic accumulator construct that accumulates *composites*, and an efficient protocol for proving knowledge of the factorization of a committed value. Based on these techniques, we developed a novel, efficient and provably secure group signature scheme.

Acknowledgements

We thank Don Coppersmith, Ivan Damgard, and Moti Yung for valuable feedback and suggestions. We also acknowledge the anonymous reviewers for Crypto'03 for their useful comments. Finally, we are grateful to Mihir Bellare and Daniele Micciancio for a preview copy of [BMW03].

References

- ACS02. J. Algesheimer, J. Camenisch, and V. Shoup. Efficient Computation Modulo a Shared Secret with Application to the Generation of Shared Safe-Prime Products. Crypto'02.
- ACJT00. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. Crypto'00.
- AST02. G. Ateniese, D. Song, and G. Tsudik. Quasi-Efficient Revocation of Group Signatures. Financial Crypto'02.
- AT99. G. Ateniese and G. Tsudik. Some Open Issues and New Directions in Group Signatures. Financial Crypto'99.
- BP97. N. Baric and B. Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. Eurocrypt'97.
- BDJR97. M. Bellare, A. Desai, E. Jorjipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. FOCS'97.
- BMW03. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction based on General Assumptions. Eurocrypt'03.
- BR93. M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. ACM CCS'93.
- B00. F. Boudot. Efficient Proof that a Committed Number lies in an Interval. Eurocrypt'00.

- BS01. E. Bresson and J. Stern. Group Signatures with Efficient Revocation. PKC'01.
- C98. J. Camenisch. Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD Thesis. ETH Zurich. 1998.
- CL02. J. Camenisch and A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. Crypto'02.
- CM98. J. Camenisch and M. Michels. A Group Signature Scheme based on an RSA-variant. Tech. Report RS-98-27, BRICS. Preliminary version appeared at Asiacrypt'98.
- CM99a. J. Camenisch and M. Michels. Separability and Efficiency for Generic Group Signature Schemes (Extended Abstract). Crypto'99.
- CS97. J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). Crypto'97.
- CGHN01. D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Nguyen. Paillier's Cryptosystem Revisited. ACM CCS'01.
- CFT98. A. Chan, Y. Frankel, and Y. Tsiounis. Each Come - Easy Go Divisible Cash. Eurocrypt'98.
- CP94. L. Chen and T. Pedersen. New Group Signature Schemes. Eurocrypt'94.
- CvH91. S. Chaum and E. van Heyst. Group Signatures. Eurocrypt'91.
- CP92. D. Chaum and T. P. Pedersen. Wallet Databases with Observers. Crypto'92.
- C96. D. Coppersmith. Finding a Small Root of a Bivariate Integer Equation; Factoring with high bits known. Eurocrypt'96.
- C03. D. Coppersmith. Personal Communication. Jan. 2003.
- D00. I. Damgard. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. Eurocrypt'00.
- DF02. I. Damgard and E. Fujisaki. An Integer Commitment Scheme Based on Groups with Hidden Order. Asiacrypt'02.
- E85. T. ElGamal, A Public-Key Cryptosystem and a Signature Scheme Based on the Discrete Logarithm, IEEE Transactions of Information Theory, 31(4), 1985, pp 469-472.
- FS86. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. Crypto'86.
- FO97. E. Fujisaki and T. Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. Crypto'97.
- G00. R. Gennaro. An Improved Pseudo-Random Generator Based on the Discrete Logarithm Problem. Crypto'00.
- GKR00. R. Gennaro, H. Krawczyk, and T. Rabin. RSA-Based Undeniable Signatures. J. Cryptology, (13)4, 2000, pp 397-416.
- KP98. J. Kilian and E. Petrank. Identity Escrow. Crypto'98.
- MR01. P. MacKenzie and M. Reiter. Two-Party Generation of DSA Signatures. Crypto'01.
- P99. P. Paillier. Public Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt'99.
- S91. C. Schnorr. Efficient Signature Generation by Smart Cards. Journal of Cryptology 4(3) 161-174, 1991.
- S01. D. Song. Practical Forward Secure Group Signature Schemes. ACM CCS'01.
- TY98. Y. Tsiounis and M. Yung. On the Security of ElGamal Based Encryption. PKC'98.
- TX03. G. Tsudik and S. Xu. Accumulating Composites and Improved Group Signing. Extended version of this paper available at <http://eprint.iacr.org/2003/112/>.

Almost Uniform Density of Power Residues and the Provable Security of ESIGN

Tatsuaki Okamoto¹ and Jacques Stern²

¹ NTT Labs, 1-1 Hikarinooka, Yokosuka-shi 239-0847 Japan
okamoto@isl.ntt.co.jp

² Dépt d'informatique, ENS, 45 rue d'Ulm
75230 Paris Cedex 05, France
Jacques.Stern@ens.fr
<http://www.di.ens.fr/~stern>

Abstract. ESIGN is an efficient signature scheme that has been proposed in the early nineties (see [14]). Recently, an effort was made to lay ESIGN on firm foundations, using the methodology of provable security. A security proof [15] in the random oracle model, along the lines of [2], appeared in support for ESIGN. However, several unexpected difficulties were found. Firstly, it was observed in [20], that the proof from [15] holds in a more restricted model of security than claimed. Even if it is quite easy to restore the usual security level, as suggested in [9], this shows that the methodology of security proofs is more subtle than it at first appears. Secondly, it was found that the proof needs the additional assumption that e is prime to $\varphi(n)$, thus excluding the case where e is a small power of two, a very attractive parameter choice. The difficulty here lies in the simulation of the random oracle, since it relies on the distribution of e -th powers, which is not completely understood from a mathematical point of view, at least when e is not prime to $\varphi(n)$. In this paper, we prove that the set of e -th power modulo an RSA modulus n , which is a product of two equal size integers p, q , is almost uniformly distributed on any large enough interval. This property allows to complete the security proof of ESIGN. We actually offer two proofs of our result: one is based on two-dimensional lattice reduction, and the other uses Dirichlet characters. Besides yielding better bounds, the latter is one new example of the use of analytic number theory in cryptography.

1 Introduction

Since the appearance of the celebrated RSA cryptosystem [18], a lot of effort has been devoted to finding alternative schemes. In the area of signature, a major challenge is to reduce the computing effort needed from the signer, since it is well known that RSA requires a full-size modular exponentiation. Among the potential candidates to answer this challenge is the ESIGN signature scheme, that has been proposed in the early nineties (see [14]). While RSA generates signatures by computing an e -th root of a hash value, ESIGN only requests to find an element whose e -th power is close enough to the hash value. Thus, the mathematical assumption underlying ESIGN is that, given an element y of \mathbb{Z}_n^* ,

it is hard to find x with e -th power lying in an interval with lower endpoint y and length say $n^{2/3}$. This is called the approximate e -th root problem, in short AERP. Combining this relaxed assumption with the use of an RSA modulus of the form $n = p^2q$ allows a very efficient way to sign, with a computing time essentially equivalent to a single exponentiation to the e -th power. This is especially attractive when e is small, and in particular a small power of two.

As most newly proposed cryptosystems, ESIGN has attracted cryptanalytic effort. Papers [3,21] described several attacks against the underlying problem, for $e = 2, 3$. Still, It is fair to say that there is no known attack against AERP when e is ≥ 4 . Recently, in connection with several standardization efforts such as IEEE P1363, Cryptrec and NESSIE, an effort was made to lay ESIGN on firm foundations, using the methodology of provable security. A security proof in the random oracle model, along the lines of [2], formally relating the security of ESIGN with the AERP problem, appeared in [15]. However, several unexpected difficulties were found. Firstly, it was observed in [20] that the proof from [15] holds in a more restricted model of security than claimed: this model, termed single occurrence chosen message attack **SO-CMA** is very similar to the usual chosen message attack scenario but does not allow the adversary to submit the same message twice for signature. This observation does not endanger the scheme in any way, and furthermore, it is quite easy to restore the usual **CMA** security, as suggested in [9]. Still, it shows that the methodology of security proofs is more subtle than it at first appears, a fact already pointed out by Shoup [19], in the context of public key encryption. Secondly, it was found that the proof needs the additional assumption that e is prime to $\varphi(n)$, thus excluding some very attractive parameter choices, notably powers of two. The difficulty here lies in the simulation of the random oracle, since it relies on the distribution of e -th powers, which is not completely understood from a mathematical point of view. In this paper, we prove that the set of e -th power modulo an RSA modulus n , which is a product of two equal size integers p, q , is almost uniformly distributed on any large enough interval. In other words, the number of e -th powers modulo n in any interval of large enough length n^δ is close to $\frac{n^\delta}{d} \frac{\varphi(n)}{n}$, where d is the number of e -th roots of unity modulo n . We actually offer two proofs of our result. The first proof relies on methods from the geometry of numbers and uses two-dimensional lattices. The second proof borrows from analytic number theory and uses Dirichlet characters and the Polya-Vinogradov inequality. Both proofs yield concrete estimates, which are enough to complete the security proof of ESIGN. Although the estimates in the second proof are sharper, we have found interesting to include the two methods, which are of independent interest.

Removing the restriction that e is prime to $\varphi(n)$ may appear a side issue. However, we believe that it is important both for practical and methodological reasons. As already noted, ESIGN has a very fast algorithm for signature generation, since its main step is a single exponentiation to the e -th power. Making e a power of two is the best way to take advantage of this feature and should be allowed by the security proof. Also, as shown by various results, notably [19,20], provable security has many subtleties. In the present paper, the

subtlety lies in the simulation of the random oracle. As far as we know, this is the only example where this part is not straightforward, and the underlying difficulty may easily be overlooked. In other words, it may appear obvious that picking x at random and suitably truncating $x^e \bmod n$ simulates a random oracle, which is the main result of our paper. However, it is not, at least when e is not prime to $\varphi(n)$ and it is actually related with deep mathematical questions of analytical number theory.

Our paper is organized as follows: we first recall some preliminaries from number theory. Next, we present the two proofs. Finally, we produce a proof of security for ESIGN, not using the assumption that e is prime to $\varphi(n)$. In this proof, we focus on the simulation of the random oracle, and explain where our result on power residues is needed.

2 Number Theoretic Preliminaries

2.1 Lattices

Let n be an RSA modulus. For any integer α , we consider the lattice

$$L(\alpha) = \{(x, y) \in \mathbb{Z}^2 \mid x - \alpha y = 0 \bmod n\}.$$

We note that $L(\alpha)$ is a two-dimensional lattice with determinant n . Thus, its shortest vector should be of euclidean norm of the order \sqrt{n} . It can be obtained by applying the Gaussian reduction algorithm. This algorithm outputs within time $\mathcal{O}((\log n)^3)$ a basis of $L(\alpha)$ consisting of two non-zero vectors $U(\alpha)$ and $V(\alpha)$ such that

$$\|U\| \leq \|V\| \text{ and } |(U, V)| \leq \|U\|^2/2,$$

where we have omitted α for clarity. From a geometrical point of view, the inequalities imply that the angle θ of U and V is such that $|\cos \theta| \leq 1/2$, hence $|\sin \theta| \geq \sqrt{3}/2$, and therefore

$$|U \wedge V| = n \geq \frac{\sqrt{3}|U||V|}{2}$$

We say that $L(\alpha)$ is an ε -good lattice if $|U|$ is bounded from below by $n^{1/2-\varepsilon}$. Note that, for such a lattice, we have

$$|V| \leq \frac{2}{\sqrt{3}}n^{1/2+\varepsilon}.$$

Lemma 1. *The number of elements α in \mathbb{Z}_n such that $L(\alpha)$ is not an ε -good lattice is at most $4n^{1-2\varepsilon}$.*

Proof. This follows from the fact that the shortest non zero vector of a lattice $L(\alpha)$ which is not ε -good lies in the disk centered at the origin, with radius $n^{1/2-\varepsilon}$. This number of integers in this disk is bounded by $4n^{1-2\varepsilon}$. To conclude, it is enough to observe that an element (x, y) of the disk other than $(0, 0)$ cannot belong to two distinct $L(\alpha)$ lattices, unless y is not in \mathbb{Z}_n^* , which cannot happen since n is an RSA integer, i.e. has two prime factors of almost equal size.

We let P be the parallelepiped spanned by U and V .

Lemma 2. *Let $L(\alpha)$ be ε -good. The width of P is at most $2n^{1/2+\varepsilon}$.*

Proof. The square of the width is indeed bounded by

$$|U|^2 + |V|^2 + 2|(U, V)| \leq 2|V|^2 + |V|^2 \leq 3|V|^2,$$

which is bounded by $4n^{1+2\varepsilon}$. The lemma follows.

Lemma 3. *Let $L(\alpha)$ be ε -good. Let I be an interval of length n^δ , with $\delta > 1/2$. The square $I \times I$ has at most $(n^{\delta-1/2} + 2n^\varepsilon)^2$ elements in $L(\alpha)$.*

Proof. let \tilde{P} be obtained by translating P by $-\frac{u+v}{2}$. We consider the set X of lattice points M such that the parallelepiped $M + \tilde{P}$ meets $I \times I$. The number of such points is clearly an upper bound for the number of lattice points inside $I \times I$. Now, the various parallelepiped $M + \tilde{P}$ are pairwise disjoint and, by lemma 2, they are contained in the square $J \times J$, obtained by enlarging I by $n^{1/2+\varepsilon}$ on each side. Summing up the areas of the individual cells, we get:

$$n|X| \leq (n^\delta + 2n^{1/2+\varepsilon})^2.$$

which provides the desired bound on the number $|X|$ of elements of X .

When $L(\alpha)$ is not ε -good, we can show a weaker bound:

Lemma 4. *Let α be any integer. Let I be an interval of length n^δ , with $\delta < 1$. The square $I \times I$ has at most $n^\delta + 1$ elements in $L(\alpha)$.*

Proof. For fixed y , there is at most one pair (x, y) such that $x - \alpha y = 0 \pmod n$ in any interval of length $< n$, such as I . This provides the requested bound $n^\delta + 1$.

2.2 Dirichlet Characters

Let G be a finite (multiplicative) abelian group. A character χ over G is a multiplicative homomorphism from G into the multiplicative group of complex numbers. The set of characters over G is a group, called the dual of G and denoted \hat{G} . Its unit χ_0 is the *principal character*, defined by $\chi_0(g) = 1$, for any $g \in G$.

The following is well-known (see [6], chapter 7):

Theorem 1. *i) There are exactly $|G|$ characters over G .
ii) For any $g \neq 1$, the following holds:*

$$\sum_{\chi \in \hat{G}} \chi(g) = 0$$

iii) For any $\chi \neq \chi_0$, the following holds:

$$\sum_{g \in G} \chi(g) = 0$$

A Dirichlet character χ is a character over \mathbb{Z}_n^* , for some integer n . The characters can be extended to all integers by using the value 0 at integers not invertible mod n . In the sequel, we will need a bound on the sum of such characters over large intervals. This is given by the Polya-Vinogradov inequality (see [5] or [6], chapter 9):

Theorem 2. *For any non principal Dirichlet character χ over \mathbb{Z}_n^* and any integer h , the following holds:*

$$\left| \sum_{x=1}^h \chi(x) \right| \leq 2\sqrt{n} \ln n.$$

Remark. When n is a prime number p , and, more generally when χ is a so-called primitive character, the multiplicative constant 2 in the above can be replaced by 1. We will not need such refinement.

3 Almost Uniform Density of e -th Powers

We now turn to our main result. We first review the standard situation of an RSA exponent.

3.1 The Case Where e Is Prime to $\varphi(n)$

Lemma 5. *Let n be an RSA modulus and e be an integer prime to $\varphi(n)$. Let I be an interval of length n^δ , with $\delta < 1$. The number of integers from I which are e -th powers of an element of \mathbb{Z}_n^* differs from $n^\delta \frac{\varphi(n)}{n}$ by at most 4.*

Proof. Since exponentiation to the e -th power is one-to-one, we have to count the number of elements in $I \cap \mathbb{Z}_n^*$. The number of multiples of p in I differs from $\frac{n^\delta}{p}$ by at most one. Similarly for q . Since there may be one multiple of pq , the final count is almost K , where

$$K = n^\delta \frac{\varphi(n)}{n}$$

and the difference with K is bounded by $3 + \frac{n^\delta}{n} \leq 4$.

We now turn to the general case. Observe that the set of e -th powers is a subgroup of \mathbb{Z}_n^* . Accordingly, we will adopt this group-theoretic setting.

3.2 A Proof Based on Lattices

We prove the following:

Theorem 3. *Let n be an RSA modulus. Let I be an interval of length n^δ , with $2/3 < \delta < 1$. Let G be any subgroup of \mathbb{Z}_n^* and let d be the number of elements of*

the quotient group \mathbb{Z}_n^*/G . Then, for some constant M , the number of elements of $I \cap G$ is $K(1 + \lambda(I))$, where

$$K = \frac{n^\delta}{d} \frac{\varphi(n)}{n}$$

and $|\lambda(I)|$ is bounded by $Mn^{1/3-\delta/2}$. Furthermore, M has the explicit bound $M \leq 5d$.

Remark. Observe that the case where $G = \mathbb{Z}_n^*$ is an easy consequence of lemma 5.

Proof. We number the elements of \mathbb{Z}_n^*/G as g_1, \dots, g_d (with g_1 the unit of G), and we let a_i be the number of elements of $\mathbb{Z}_n^* \cap I$ which equal g_i modulo G . We first show an upper bound for

$$A = \sum_{i=1}^d a_i^2$$

For any pair (x, y) in $I \times I$, we define $\sigma(x, y)$ as $xy^{-1} \bmod n$, when x, y both belong to \mathbb{Z}_n^* and set $\sigma(x, y) = \infty$ otherwise. Observe that A can be interpreted as the number of elements (x, y) of $\mathbb{Z}_n^* \cap I$ such that $\sigma(x, y) \in G$. Indeed, $xy^{-1} \bmod n$ is in G if and only if x and y are equal modulo G . We now use a counting argument to estimate the size of $\sigma^{-1}(\alpha)$, when α ranges over G . We distinguish two cases

1. When $L(\alpha)$ is an ε -good lattice, then, by lemma 3, $\sigma^{-1}(\alpha)$ has at most $(n^{\delta-1/2} + 2n^\varepsilon)^2$ elements.
2. Otherwise, we use lemma 4 to get that $\sigma^{-1}(\alpha)$ has at most $n^\delta + 1$ elements, which we replace by the (crude) bound $2n^\delta$.

Since there are at most $4n^{1-2\varepsilon}$ values of α which give rise to a lattice $L(\alpha)$ which is not ε -good, we get

$$A \leq \frac{\varphi(n)}{d} (n^{\delta-1/2} + 2n^\varepsilon)^2 + 8n^{1-2\varepsilon+\delta}.$$

Upperbounding $\varphi(n)$ by n , we get:

$$A \leq \frac{n^{2\delta}}{d} (1 + 2n^{1/2+\varepsilon-\delta})^2 + 8n^{1-2\varepsilon+\delta}.$$

We now set $\varepsilon = 1/6$. This yields the bound

$$A \leq \frac{n^{2\delta}}{d} (1 + 2n^{2/3-\delta})^2 + 8n^{2\delta} n^{2/3-\delta}.$$

Since δ is $> 2/3$, $n^{2/3-\delta}$ is < 1 and its square is bounded by $n^{2/3-\delta}$. We finally get:

$$A \leq \frac{n^{2\delta}}{d} (1 + (8 + 8d)n^{2/3-\delta}).$$

We now use the fact that the sum $B = \sum_{i=1}^d a_i$ is essentially known. Referring to the proof of lemma 5 above, we see that it differs from

$$n^\delta \frac{\varphi(n)}{n}$$

by at most 4. Now, the vector (a_1, \dots, a_d) lies on the d -dimensional hyperplane H defined by $B = \sum_{i=1}^d x_i$. Let (b_1, \dots, b_d) be the orthogonal projection of the origin on H . It is easily seen that $b_i = B/d$. The square of the euclidean distance between (a_1, \dots, a_d) and (b_1, \dots, b_d) is $\sum_{i=1}^d a_i^2 + \sum_{i=1}^d b_i^2 - 2 \sum_{i=1}^d a_i b_i$. This is $A - \frac{B^2}{d}$. we are thus led to find a lower bound for $\frac{B^2}{d}$. Using the same estimate as for the proof of lemma 5, we write

$$\frac{B^2}{d} \geq \frac{n^{2\delta}}{d} \left(\frac{\varphi(n)}{n} - 4n^{-\delta} \right)^2.$$

Using the fact that we have an RSA modulus, we use the lower bound $1 - \frac{3}{\sqrt{n}}$ for $\frac{\varphi(n)}{n}$ and, combining with the above, obtain the final bound

$$\frac{B^2}{d} \geq \frac{n^{2\delta}}{d} \left(1 - \frac{14}{\sqrt{n}} \right).$$

Finally, piecing bounds together, we get:

$$A - \frac{B^2}{d} \leq \frac{n^{2\delta}}{d} (22 + 8d) n^{2/3-\delta},$$

which provides a bound for $(a_1 - b_1)^2 = (a_1 - B/d)^2$. Observing that we only have to deal with $d \geq 2$, we easily get that $|a_1 - B/d|$ is at most

$$\sqrt{19} n^\delta n^{1/3-\delta/2}.$$

Replacing B/d by the constant

$$K = \frac{n^\delta}{d} \frac{\varphi(n)}{n},$$

yields a minute difference $\leq 4/d$, which we handle by slightly raising the $\sqrt{19}$ constant. Thus, a_1 can be written $K(1 + \lambda(I))$, with

$$|\lambda(I)| \leq (\sqrt{19} + \gamma) d \frac{n}{\varphi(n)} n^{1/3-\delta/2},$$

We finally handle the term $\frac{n}{\varphi(n)}$ by raising the constant again. This gives the requested bound

$$|\lambda(I)| \leq 5dn^{1/3-\delta/2}.$$

3.3 A Proof Based on Characters

We now show that a better bound for $\lambda(I)$, can be obtained as a consequence of the Polya-Vinogradov inequality of theorem 2.

Theorem 4. *Let n be an RSA modulus. Let I be an interval of length n^δ , with $1/2 < \delta < 1$. Let G be any subgroup of \mathbb{Z}_n^* and let d be the number of elements of the quotient group \mathbb{Z}_n^*/G . Then, for some constant M , the number of elements of $I \cap G$ is $K(1 + \lambda(I))$, where*

$$K = \frac{n^\delta}{d} \frac{\varphi(n)}{n},$$

and $|\lambda(I)|$ is bounded by $Mn^{1/2-\delta} \ln n$. Furthermore, M has the explicit bound $M \leq 5d$.

Proof. We consider the dual \hat{H} of the quotient group $H = \mathbb{Z}_n^*/G$. For any character χ over H , we can extend χ to G , by composing with the canonical homomorphism from G onto H . We still denote by χ , the resulting character. Since there are d characters altogether, we get, using the relations in theorem 1, that the number of elements of $I \cap G$ is equal to the sum

$$\frac{1}{d} \sum_{x \in I} \sum_{\chi \in \hat{G}} \chi(x),$$

Changing the order of the sums, we see that this number consists of two terms:

1. one comes from the principal character and equals: $\frac{|I \cap \mathbb{Z}_n^*|}{d}$,
2. the others come from the non trivial characters, and, by the Polya-Vinogradov inequality, each is bounded by $\frac{4}{d} n^{1/2} \ln n$.

By lemma 5, the first contribution differs from

$$K = \frac{n^\delta}{d} \frac{\varphi(n)}{n}$$

by at most $\frac{4}{d}$. Summing up with the second contribution, we obtain the bound:

$$\frac{4}{d} + \frac{4(d-1)}{d} n^{1/2} \ln n \leq 4n^{1/2} \ln n.$$

Altogether, we obtain that the number of elements of $I \cap G$ is $K(1 + \lambda(I))$, with

$$\lambda(I) \leq 4d \frac{n}{\varphi(n)} n^{1/2-\delta} \ln n,$$

Using the fact that n is an RSA modulus, we estimate $\varphi(n)$, by $n(1 - 4/\sqrt{n})$, and bound the multiplicative constant by a term

$$\simeq 4d \left(1 + \frac{4}{\sqrt{n}}\right).$$

This is bounded by 5. The result follows.

It should be noted that an even better bound has been obtained by Burgess [4]. The bound covers the case $1/4 < \delta < 1$, and reads:

$$|\lambda(I)| \leq Mdn^{\frac{1}{4r} - \frac{\delta}{r+1}} \ln n,$$

for any positive r . However, the constant M is not not explicit, and therefore the improvement is not well suited for our purposes.

4 The Security Proof of ESIGN

In this section, we review the proof of security for ESIGN in view of the previous results. For the reader's convenience, we first provide a short description of the scheme and of the underlying mathematical problem AERP. We follow [15].

4.1 Description

The key generation algorithm of ESIGN chooses two large primes p, q of equal size k and computes the modulus $n = p^2q$. The sizes of p, q are set in such a way that the binary length $|n|$ of n equals $3k$. Additionally, an exponent $e > 4$ is chosen, possibly a small power of 2.

Signature generation uses a hash function \mathcal{H} , outputting strings of length $k - 1$, and is performed as follows:

1. Pick at random r in \mathbb{Z}_{pq}^* .
2. Convert $(0\|\mathcal{H}(m)\|0^{2k})$ into an integer y and compute $z = (y - r^e) \bmod n$.
3. Compute

$$w_0 = \left\lceil \frac{z}{pq} \right\rceil$$

$$w_1 = w_0.pq - z$$

4. If $w_1 \geq 2^{2k-1}$, return to step 1.
5. Set $u = w_0.(er^{e-1})^{-1} \bmod p$ and $s = r + upq$.
6. Output s as the signature of m .

Signature verification converts integer $s^e \bmod n$ into a bit string S of length $3k$ and checks that $[S]^k = 0\|\mathcal{H}(m)$, where $[S]^k$ denotes the k leading bits of S .

The key idea in ESIGN is that the arithmetical progression $r^e \bmod n + tpq$ consists of e -th powers of integers easily computed from r . The signature generation algorithm simply adjusts t so as to fall into a prescribed interval, with lower end-point y . The test at step 4 actually sets the length of this prescribed interval to 2^{2k-1} .

The following lemma will prove useful in the sequel.

Lemma 6. *For a fixed message m , the e -th power $s^e \bmod n$ of the output s of the signature generation algorithm is uniformly distributed over the set of e -th powers of elements of \mathbb{Z}_n^* lying in the interval $[y, y + 2^{2k-1})$.*

Proof. Denote by $S(y)$ the intersection of the set of e -th powers in \mathbb{Z}_n^* and the interval $[y, y + 2^{2k-1})$. Observe that $s = r + tpq$ uniquely defines $r = s \bmod pq$ from s . This shows that any element in $S(y)$ comes from a single r . To see that all elements in $S(y)$ are uniformly hit, pick $w \in S(y)$, consider any r in \mathbb{Z}_{pq}^* such that $r^e = w \bmod pq$, and apply the signature generation algorithm with r , disregarding the check at step 4. This produces a value of s such that $s^e = r^e = w \bmod pq$. Thus, w and $s^e \bmod n$ lie in the arithmetical progression $s^e + tpq$. Since this arithmetical progression has a single element in the interval $[y, y + 2^{2k-1})$, we get that $s^e \bmod n = w$. The check at step 4 turns out correct and the signature generation algorithm duly hits w as many times as the number of e -th roots of an e -th power.

4.2 The Approximate e -th Root Problem

As noted in the previous section, RSA moduli of the form p^2q offer a very efficient way to solve the following problem, having knowledge of the factorization of n : given n and y in \mathbb{Z}_n^* , find x such that $x^e \bmod n$ lies in the interval $[y, y + 2^{2k-1})$, where the bit-size of n is $3k$ and $[y, y + 2^{2k-1})$ denotes $\{u | y \leq u < y + 2^{2k-1}\}$.

It is conjectured that the above problem, called the approximate e -th root problem (AERP) in [15], is hard to solve. More precisely, denote by $\text{Succ}^{\text{aerp}}(\tau, k)$ the probability for any adversary \mathcal{A} to find an element whose e -th power lies in the prescribed interval, within time τ , in symbols:

$$\Pr[(n, e) \leftarrow \mathcal{K}(1^k), y \leftarrow \mathbb{Z}_n, x \leftarrow \mathcal{A}(n, e, y) : (x^e \bmod n) \in [y, y + 2^{2k-1})],$$

then, for large enough moduli, this probability is extremely small. Variants of the above can be considered, where the length of the interval is replaced by 2^{2k} or 2^{2k+1} .

4.3 Security Proof

We now complete the security proof of ESIGN, in order to cover the case where e is not prime to $\varphi(n)$. We use the random oracle model and prove the following security result, where $T_{\text{exp}}(k)$ denotes the computing time of modular exponentiation modulo a $3k$ -bit integer.

Theorem 5. *Let \mathcal{A} be a SO-CMA-adversary against the ESIGN signature scheme that produces an existential forgery, with success probability ε , within time τ , making q_H queries to the hash function and q_s distinct requests to the signing oracle respectively. Then, AERP can be solved with probability ε' , and within time τ' , where*

$$\begin{aligned} \varepsilon' &\geq \frac{\varepsilon - 2^{-k+1}}{q_H} - (q_H + q_s) \times (3/4)^k - \frac{ke^2(q_H + q_s)}{2^{k-6}} \\ \tau' &\leq \tau + k(q_s + q_H) \cdot T_{\text{exp}}(k). \end{aligned}$$

Our method of proof is inspired by Shoup [19]. It differs from [15] but extends the proof given in [20]. The security estimates are similar and show the same multiplicative loss q_H : contrary to schemes based on self-reducible problems, it does not seem that this can be avoided. Recall that earlier proofs used the assumption that e is prime to $\varphi(n)$, which we avoid. This brings additional terms in the security estimates, which account for the simulation of the random oracle. Also note that our security model is the single occurrence chosen message attack SO-CMA from [20], where the attacker is only allowed to query each message once. As already noted, it is easy to modify the scheme to withstand CMA attackers and our proof can be modified accordingly.

As usual, the proof considers a sequence of **Game**₁, **Game**₂, etc of modified attack games starting from the actual game **Game**₀. Each of the games operates on the same underlying probability space, only the rules defining how the view is computed differ from game to game.

Proof. (of Theorem 5). We consider an adversary \mathcal{A} outputting an existential forgery (m, s) , with probability ε , within time τ . We denote by q_H and q_s respectively the number of queries from the random oracle \mathcal{H} and from the signing oracle. As explained, we start by playing the game coming from the actual adversary, and modify it step by step, until we reach a final game, whose success probability has an upper-bound obviously related to solving AERP on a random instance (n, e, v) .

Game₀: The key generation algorithm $\mathcal{K}(1^k)$ is run and produces a pair of keys (pk, sk) . The adversary \mathcal{A} is fed with pk and, querying the random oracle \mathcal{H} and the signing oracle Σ_{sk} , it outputs a pair (m, s) . We denote by S_0 the event that $V_{pk}(m, s) = 1$. We use a similar notation S_i in any **Game** _{i} below. By definition, we have

$$\Pr[S_0] = \varepsilon.$$

Game₁: In this game, we discard executions, which end up outputting a valid message/signature pair (m, s) , such that m has not been queried from \mathcal{H} . This means restricting to the event **AskH** that m has been queried from \mathcal{H} . Unwinding the ESIGN format, we write: $s^e = 0 \parallel w \parallel \star \bmod n$. If **AskH** does not hold, $\mathcal{H}(m)$ is undefined, and the probability that $\mathcal{H}(m) = w$ holds is $1/2^{k-1}$: $\Pr[S_0 \mid \neg \text{AskH}] \leq 2^{-k+1}$. Thus,

$$\Pr[S_1] = \Pr[S_0 \wedge \text{AskH}] \geq \Pr[S_0] - 2^{-k+1}.$$

Game₂: In this game, we choose at random an index κ between 1 and q_H . We let m_κ be the κ -th message queried to \mathcal{H} by the adversary. We then discard executions which output a valid message/signature pair (m, s) , such that $m \neq m_\kappa$. Since the additional random value κ is chosen independently of the execution of **Game**₁,

$$\Pr[S_2] = \Pr[S_1]/q_H.$$

Game₃: In this game, we immediately abort if a signing query involves message m_κ . By the definition of existential forgery, this only eliminates executions outside S_2 . Thus:

$$\Pr[S_3] = \Pr[S_2].$$

Game₄: We now simulate the random oracle \mathcal{H} , by maintaining an appropriate list, which we denote by **H-List**. For any fresh query m , other than the κ -th query, we pick at random $u \in \mathbb{Z}_n$ and compute $z = u^e \bmod n$, until the most significant bit of z is 0. We next parse z as $0 \| w \| \star$, where w is of length $k-1$ and check whether $z - w \cdot 2^{2k}$ is less than 2^{2k-1} . If this is true, we store (m, u, w) in **H-List** and returns w as the answer to the oracle call. Otherwise we restart the simulation of the current query. From theorem 4, we see that the game differs from the previous due to a slightly biased simulated distribution. This distribution is obtained by setting $z = w2^{2k}$, counting the number of e -th powers of elements of \mathbb{Z}_n^* lying in the interval $[z, z + 2^{2k-1})$, and multiplying by a suitable constant for normalisation. Recall that, an element x of $[z, z + 2^{2k-1})$ is an e -th power modulo n if and only if $x \bmod pq$ is an e -th power modulo pq . This is basically a restatement of the key idea of ESIGN. Thus, setting $z' = z \bmod pq$, we have to count the number $\nu(z)$ of elements of the interval $[z', z' + 2^{2k-1})$, which belong to the subgroup G of e -th powers in \mathbb{Z}_{pq}^* . By theorem 4, the result is $K(1 + \lambda(z))$, where $|\lambda(z)|$ is bounded by $M(pq)^{1/2-2k+1} \ln pq$, and where K, M are appropriate constants. This yields

$$|\lambda(z)| \leq M2^{-k+1/2} \ln pq$$

Upperbounding $\ln pq$ by $3/2 \log pq$, we get:

$$|\lambda(z)| \leq 3Mk2^{-k+1/2}$$

Now, it is easily seen that any probability distribution obtained by normalizing a function $\nu(z) = K(1 + \lambda(z))$, where $\lambda(z)$ is bounded by λ , differs from the uniform distribution by at most $\frac{2\lambda}{1-\lambda} \simeq 2\lambda$. Taking into account the bound $M \leq 5d$, where d is the number of elements of the quotient of \mathbb{Z}_{pq}^* by the subgroup of e -th powers, and bounding d by e^2 , we conclude that the statistical distance of the simulated distribution to the uniform distribution is bounded by twice the bound on λ , which is $30e^2k2^{-k+1/2} \leq 64e^2k2^{-k}$. Summing up for all oracle calls, we get:

$$| \Pr[S_4] - \Pr[S_3] | \leq \frac{ke^2(q_H + q_s)}{2^{k-6}}.$$

Game₅: Here, we modify the previous simulation stopping and aborting the game when the \mathcal{H} query cannot be simulated after k trials. This game differs from the previous one when w remains undefined after k attempts.

$$\Pr[S_5] \geq \Pr[S_4] - (q_H + q_s) \times (3/4)^k.$$

Game₆: We complete the simulation by replacing $\mathcal{H}(m_\kappa)$ by v , where v is an additional random string, which serves as an input to the AERP problem. The distribution of \mathcal{H} -outputs is unchanged:

$$\Pr[S_6] = \Pr[S_5].$$

Game₇: We finally simulate the signing oracle: for any m , whose signature is queried, we know that $m = m_\kappa$ does not hold, since corresponding executions have been aborted in **Game₃**. Thus $\mathbf{H}\text{-List}$ includes a triple (m, u, w) , such that $u^e \bmod n$ has its k leading bits of the form $0 \| \mathcal{H}(m)$. Accordingly, u provides a valid signature of m . Furthermore, referring to lemma 6, we see that the signing oracle outputs a value s , such that $s^e \bmod n$ is uniformly distributed over all elements of \mathbb{Z}_n^* whose $k+1$ leading bits match up with $0 \| \mathcal{H}(M) \| 0$. Keeping in mind that $\mathcal{H}(m)$ is chosen at random, we conclude that s and u follow an identical distribution. We now argue that the simulation is perfect. The key fact is that, due to the **SO-CMA** setting, all inputs m submitted to the \mathcal{H} oracle by the signing oracle during execution are distinct. This implies that the values of s returned at each invocation of the signing oracle are independent. Since the values of u are also independent, the overall distribution of simulated signatures obtained at **Game₇** is identical to the distribution of actual signatures from **Game₆**. Therefore,

$$\Pr[S_7] = \Pr[S_6].$$

Summing up the above inequalities, we obtain

$$\begin{aligned} \Pr[S_7] &\geq \Pr[S_4] - (q_H + q_s) \times \left(\frac{3}{4}\right)^k \geq \Pr[S_3] - (q_H + q_s) \times \left(\frac{3}{4}\right)^k - \frac{ke^2(q_H + q_s)}{2^{k-6}} \\ &\geq \frac{\Pr[S_1]}{q_H} - (q_H + q_s) \times \left(\frac{3}{4}\right)^k - \frac{ke^2(q_H + q_s)}{2^{k-6}} \\ &\geq \frac{\varepsilon - 2^{-k+1}}{q_H} - (q_H + q_s) \times \left(\frac{3}{4}\right)^k - \frac{ke^2(q_H + q_s)}{2^{k-6}} \end{aligned}$$

When **Game₇** terminates outputting a valid message/signature pair (m, s) , we unwind the **ESIGN** format and get $s^e = (0 \| v \| \star) \bmod n$, with $v = \mathcal{H}(m)$. If S_7 holds, we know that $m = m_\kappa$ and $\mathcal{H}(m) = v$. This leads to an element whose e -th power lies in the interval $[v2^{2k}, v2^{2k} + 2^{2k})$, thus solving an instance of **AERP**. We finally have: $\Pr[S_7] \leq \text{Succ}^{\text{aerp}}(\tau', k)$, where τ' denotes the running time of **Game₇**. This is the requested bound. Observe that τ' is the sum of the time for the original attack, plus the time required for simulations, which amounts to at most $k(q_s + q_H)$ modular exponentiations.

Remark. The security proof that appears in [15] replaces the k multiplicative factor in the running time by 4. This is intuitively related to the fact that, on average, it takes at most 4 steps to perform the simulation of each call to \mathcal{H} in **Game₄**. It is actually possible to improve our time estimate

$$\tau' \leq \tau + k(q_s + q_H) \cdot T_{\text{exp}}(k),$$

to

$$\tau' \leq \tau + 4(q_s + q_H) \cdot T_{exp}(k),$$

This uses a method due to Jonsson [12]. It modifies the strategy for the simulation of \mathcal{H} in Game_5 : instead of limiting the number of trials allowed, at each execution, to find a value of z in the correct range, it sets a counter that bounds the overall number of retries, during the entire algorithm.

References

1. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73, ACM Press, New York, 1993.
2. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399–416, Springer-Verlag, Berlin, 1996.
3. E. Brickell and J. M. DeLaurentis. An Attack on a Signature Scheme proposed by Okamoto and Shiraishi. In *Crypto '85*, LNCS 218, pages 28–32, Springer-Verlag, Berlin, 1986.
4. D.A. Burgess. On character sums and primitive roots, *Proc. London Math. Soc.*, 12 (1962), 179–192.
5. H. Davenport. *Multiplicative Number theory*, Graduate Texts in Mathematics, Vol 74, Springer Verlag, (1980).
6. W.J. Ellison and M. Mendes France. *Les nombres premiers*, Hermann, Paris (1975).
7. M. Girault, P. Toffin and B. Vallée. Computation of Approximate L-th Roots Modulo n and Application to Cryptography. In *Crypto '88*, LNCS 403, pages 100–118, Springer-Verlag, Berlin, 1989.
8. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
9. L. Granboulan. How to repair ESIGN, NESSIE internal document, may 2002. See <http://www.cryptonessie.org>, Docuemnyt NES/DOC/ENS/WP5/019.
10. IEEE Standard 1363–2000. Standard Specifications for Public Key Cryptography. IEEE. Available from <http://grouper.ieee.org/groups/1363>, August 2000.
11. IEEE P1363a Draft Version 9. Standard Specifications for Public Key Cryptography: Additional Techniques.
12. J. Jonsson. Security Proofs for RSA–PSS and Its Variants. Cryptology ePrint Archive 2001/053. June 2001. Available from <http://eprint.iacr.org/>.
13. A. K. Lenstra, H. W. Lenstra and L. Lovász, Factoring polynomials with rational coefficients, *Mathematische Ann.*, 261, (1982), 513–534.
14. T. Okamoto. A Fast Signature Scheme Based on Congruential Polynomial Operations. *IEEE Transactions on Information Theory*, IT–36 (1), pages 47–53, 1990.
15. T. Okamoto, E. Fujisaki and H. Morita. TSH-ESIGN: Efficient Digital Signature Scheme Using Trisection Size Hash, Submission to P1363a, 1998.
16. T. Okamoto and A. Shiraishi. A Fast Signature Scheme Based on Quadratic Inequalities. *Proc. of the ACM Symp. Security and Privacy*, ACM Press, pages 123–132, 1985.
17. G. Pólya. Über die Verteilung des quadratischen Reste und Nichtreste, *Göttinger Nachrichten* (1918), 21–26.

18. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
19. V. Shoup. OAEP Reconsidered. In *Crypto '2001*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001. Also appeared in the Cryptology ePrint Archive 2000/060. November 2000. Available from <http://eprint.iacr.org/>.
20. J. Stern, D. Pointcheval, J. Malone-Lee, and N. Smart. Flaws in Applying Proof Methodologies to Signature Schemes. In *Crypto '02*, LNCS 2442, pages 93–110.
21. B. Vallée, M. Girault, and P. Toffin. How to break Okamoto's Cryptosystem by Reducing Lattice Bases. In *Eurocrypt '88*, LNCS 330, pages 281–292, Springer-Verlag, Berlin, 1988.
22. B. Vallée, M. Girault and P. Toffin. How to Guess ℓ th Roots Modulo n by Reducing Lattice Bases. In *AAECC-6*, LNCS 357, pages 427–442, Springer-Verlag, Berlin, 1988.
23. I.M. Vinogradov. Sur la distributions des résidus et des non-résidus des puissances, *J. Phys.-Math. Soc. Perm.* 1 (1918), 94-96.

Rotations and Translations of Number Field Sieve Polynomials

Jason E. Gower*

CERIAS and Department of Mathematics
Purdue University, West Lafayette
IN 47907-2067, USA
`jgower@math.purdue.edu`

Abstract. We present an algorithm that finds polynomials with many roots modulo many primes by rotating candidate Number Field Sieve polynomials using the Chinese Remainder Theorem. We also present an algorithm that finds a polynomial with small coefficients among all integral translations of X of a given polynomial in $\mathbb{Z}[X]$. These algorithms can be used to produce promising candidate Number Field Sieve polynomials.

1 Introduction

The Number Field Sieve (NFS) [1] is the fastest (asymptotically) known general integer factorization algorithm. When attempting to factor an integer N with NFS, we must first choose a polynomial $f \in \mathbb{Z}[X]$ with a known root m modulo N . When f has many roots modulo many small primes, then we say f has good *root properties*. If the magnitude of values taken by f are small, then we say that f has small *size*. It can be shown (heuristically) that if f has good root properties and has small size, then NFS should run faster than when f does not have these properties.

Procedures for generating candidate NFS polynomials with good root properties and small size are described in [2]. Specifically, through the use of *rotations* and *translations*, we hope to generate polynomials with better than average root properties and size. In Sect. 2 we recall some basic facts about homogeneous polynomials and their roots modulo primes. In Sect. 3 we then recall the standard method for generating candidate NFS polynomials. In Sect. 4 we describe a method for rotating candidate NFS polynomials to generate new candidate NFS polynomials with many distinct roots modulo many primes. We discuss how to find potentially small polynomials among polynomials of the form $f(X - \alpha)$, where $f \in \mathbb{Z}[X]$ is fixed and $\alpha \in \mathbb{Z}$ in Sect. 5. We present an algorithm in Sect. 6 that finds candidate NFS polynomials with good root properties and small size based on the methods discussed in Sect. 3-5. Finally, we conclude in Sect. 7 with a discussion of how “good” candidate NFS polynomials generated by the algorithms presented in this paper should be.

* This work was supported in part by grants from the CERIAS Center at Purdue University and from the Lily Endowment Inc.

2 Root Properties

Suppose $f = a_d X^d + \cdots + a_0 \in \mathbb{Z}[X]$ is a polynomial of degree d and $p \in \mathbb{Z}$ is prime. The *homogenization* of f is the polynomial $F \in \mathbb{Z}[X, Y]$ defined by $F(X, Y) = Y^d f(X/Y)$. A co-prime pair (a, b) is a *root* of F modulo p if $F(a, b) \equiv 0 \pmod{p}$. We shall sometimes refer to (a, b) as simply a root of F if the prime p is understood. Thinking of (a, b) as a point on the projective line $\mathbb{P}^1(\mathbb{F}_p)$, we follow the language of [2] and divide roots into two classes:

- **Projective Roots:** A root (a, b) where p divides b is called *projective*. Note that F will have projective roots if and only if p divides a_d .
- **Regular Roots:** A root (a, b) where p does not divide b is called *regular*. Here, (a, b) is a regular root iff $f(ab^{-1}) \equiv 0 \pmod{p}$, where b^{-1} is calculated in \mathbb{F}_p . A regular root (a, b) with $p \mid a$ is sometimes called a *zero* root.

3 Base- m Method

Given positive integers m, N with $m \leq N$, it is not difficult to find a polynomial $f \in \mathbb{Z}[X]$ such that $f(m) \equiv 0 \pmod{N}$. A well-known method for doing this is the *base- m method* described in [1]. If $N = a_d m^d + \cdots + a_0$ is the base- m representation of N , where $0 \leq a_i < m$, then by taking $f(X) = a_d X^d + \cdots + a_0$ we have $f(m) \equiv 0 \pmod{N}$. Given d , the degree of f can be chosen to be d by taking

$$\left\lfloor N^{\frac{1}{d+1}} \right\rfloor < m \leq \left\lfloor N^{\frac{1}{d}} \right\rfloor$$

and constructing f as above. Furthermore, suppose we want to construct a polynomial with leading coefficient L and degree d . If $1 \leq L < N^{1/(d+1)} - 1$, then it is not hard to see that a base- m polynomial with

$$\left\lfloor \left(\frac{N}{L+1} \right)^{\frac{1}{d}} \right\rfloor < m \leq \left\lfloor \left(\frac{N}{L} \right)^{\frac{1}{d}} \right\rfloor$$

will have leading coefficient $a_d = L$.

Finally, we can arrange $- \lfloor m/2 \rfloor < a_i \leq \lfloor m/2 \rfloor$ for $0 \leq i < d$ by using the transformation

$$\begin{aligned} &\text{if } a_i > \lfloor m/2 \rfloor, \text{ then} \\ &\quad a_i \leftarrow a_i - m \\ &\quad a_{i+1} \leftarrow 1 + a_{i+1} \end{aligned}$$

for $i = 0, 1, \dots, d-1$. It should be noted that this transformation may change the leading coefficient. This happens precisely when $a_{d-1} > \lfloor m/2 \rfloor$, after applying the transformation. If $a_{d-1} \approx \lfloor m/2 \rfloor$, then $|a_{d-1}| \approx |a_{d-1} - m|$ so we can leave a_{d-1} alone; otherwise, we can change the value of m and start over.

We summarize the above with the following algorithm:

Algorithm 1. (Modified base- m method) Let i, d, L , and N be positive integers with $1 \leq L < N^{1/(d+1)} - 1$. This algorithm attempts to find an integer m and a polynomial $f = a_d X^d + \cdots + a_0 \in \mathbb{Z}[X]$ with $a_d = L$ and $|a_j| \leq m/2$, for $0 \leq j < d - 1$, such that $f(m) \equiv 0 \pmod{N}$. The parameter i allows the user to vary the value of m .

1. [Generate m] Set $m \leftarrow i + \left\lfloor \left(\frac{N}{L+1} \right)^{\frac{1}{d}} \right\rfloor$. If $m > \left\lfloor \left(\frac{N}{L-1} \right)^{\frac{1}{d}} \right\rfloor$, then print “ i is too big” and terminate the algorithm.
2. [Build base- m representation of N] Set $temp \leftarrow N$. For $j = 0, \dots, d$, do
 $a_j \leftarrow temp \bmod m$
 $temp \leftarrow (temp - a_j)/m$.
3. [Adjust a_j] For $j = 0, 1, \dots, d - 2$, do
 If $a_j > \lfloor m/2 \rfloor$, then
 $a_j \leftarrow a_j - m$
 $a_{j+1} \leftarrow 1 + a_{j+1}$.
4. [Build polynomials] Set
 $f_1(X) \leftarrow a_d X^d + \cdots + a_0$.
 If $a_{d-1} > \lfloor m/2 \rfloor$ then set
 $a_{d-1} \leftarrow a_{d-1} - m$
 $a_d \leftarrow 1 + a_d$
 $f_2(X) \leftarrow a_d X^d + \cdots + a_0$;
 otherwise set
 $f_2(X) \leftarrow f_1(X)$.
5. [Output and Terminate] If the leading coefficient of $f_2(X)$ is L , then return m and $f_2(X)$ and terminate the algorithm. Otherwise, if the leading coefficient of $f_1(X)$ is L , then return m and $f_1(X)$ and terminate the algorithm. Finally, if neither leading coefficient is L , then print “ i is too big” and terminate the algorithm.

Note that the homogenization of the polynomial generated by Algorithm 1 will have projective roots modulo each prime dividing L .

4 Rotations

Suppose $f \in \mathbb{Z}[X]$ is a polynomial of degree d with root m modulo N . Then $g = f + (b_r X^r + \cdots + b_0)(X - m)$, with $0 \leq r < d$, is a polynomial of degree d (unless $r = d - 1$ and $b_{d-1} = -a_d$) with root m modulo N . We call the polynomial $b_r X^r + \cdots + b_0$ a *rotation* of f . Given a finite set of powers of distinct primes \mathcal{S} , we look for a rotation that yields a polynomial with good root properties with respect to \mathcal{S} . In [2], linear rotations ($r = 1$) are found using a sieve-like procedure. We present an algorithm that finds promising higher degree rotations using the Chinese Remainder Theorem (CRT). The basic idea is to first choose roots $k_{ij} \bmod p_i^{e_i}$. Then for each i find a rotation that yields a polynomial with roots $k_{ij} \bmod p_i^{e_i}$, and finally use CRT to find a single rotation that yields a polynomial with roots k_{ij} for all i, j .

Suppose $\mathcal{S} = \{p_1^{e_1}, \dots, p_s^{e_s}\}$, where $p_i \neq p_j$ unless $i = j$, and $e_i \geq 1$ for all i . For each p_i and each $0 \leq j \leq r$, choose k_{ij} such that $0 \leq k_{ij} < p_i$, $k_{ij} \neq k_{il}$ unless $j = l$, and p_i does not divide $m - k_{ij}$. This requires $r \leq p_i - 2$ for all i . Now set $z_{ij} = (m - k_{ij})^{-1} f(k_{ij}) \bmod p_i^{e_i}$. If we set $g = f + (b_r X^r + \dots + b_0)(X - m)$, then k_{ij} will be a root of g modulo $p_i^{e_i}$ for $0 \leq j \leq r$ if

$$b_r k_{ij}^r + \dots + b_1 k_{ij} + b_0 \equiv z_{ij} \bmod p_i^{e_i}.$$

To determine the b_i modulo $p_i^{e_i}$, we must solve the matrix congruence

$$\begin{pmatrix} 1 & k_{i0} & k_{i0}^2 & \dots & k_{i0}^r \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & k_{ir} & k_{ir}^2 & \dots & k_{ir}^r \end{pmatrix} \begin{pmatrix} b_0 \\ \vdots \\ b_r \end{pmatrix} \equiv \begin{pmatrix} z_{i0} \\ \vdots \\ z_{ir} \end{pmatrix} \bmod p_i^{e_i}. \quad (1)$$

We have chosen the k_{ij} so that we may solve this system uniquely. Let $(b_{i0}, \dots, b_{ir})^T$ denote the unique solution vector modulo $p_i^{e_i}$. Finally, we solve the system of linear congruences

$$\begin{aligned} b_j &\equiv b_{1j} \bmod p_1^{e_1} \\ b_j &\equiv b_{2j} \bmod p_2^{e_2} \\ &\vdots \\ b_j &\equiv b_{sj} \bmod p_s^{e_s} \end{aligned}$$

using CRT, for each $0 \leq j \leq r$.

We now have a polynomial $g = f + (b_r X^r + \dots + b_0)(X - m)$ such that $g(k_{ij}) \equiv 0 \bmod p_i^{e_i}$ for $0 \leq j \leq r$ and $1 \leq i \leq s$. We should note that the coefficients of g may be larger than the coefficients of f . Explicitly, if $f = a_d X^d + \dots + a_0$, then $g = c_d X^d + \dots + c_0$, where

$$c_i = \begin{cases} a_d + b_{d-1} & \text{if } i = d \\ a_i + b_{i-1} - m b_i & \text{if } 1 \leq i < d \\ a_0 - m b_0 & \text{if } i = 0 \end{cases}, \quad (2)$$

where $b_i = 0$ if $i > r$.

Now let $C = \prod_{i=1}^s p_i^{e_i}$. We would usually take the b_i as the least positive residue modulo C , but it should be noted that we may as well take $b_i + lC$, where $l \in \mathbb{Z}$, if it suits our purposes. In the best case scenario, we can choose the b_i so that g is a skewed polynomial with coefficients that grow geometrically (roughly) from c_d to c_0 . If this is not the case, then it may be possible by using a suitable translation (see Sect. 5). Finally we note that if f has many roots modulo many primes, then its homogenization F will have many regular roots modulo many primes.

We summarize this discussion with the following algorithm:

Algorithm 2. (Rotation) Let $f \in \mathbb{Z}[X]$ be a polynomial of degree d , with root m modulo N . Let \mathcal{S} be a finite set of powers of distinct primes $\mathcal{S} = \{p_1^{e_1}, \dots, p_s^{e_s}\}$

and $0 \leq r < d$. This algorithm finds a polynomial $g \in \mathbb{Z}[X]$ with root m modulo N and at least $r + 1$ distinct roots modulo each $p_i^{e_i} \in \mathcal{S}$. If $r = d - 1$, then the degree of g will be either d or $d - 1$; otherwise the degree of g will be d .

1. [Check parameters] Order the primes so that $p_1 < p_2 < \dots < p_s$. If $r > p_1 - 2$, then print “Either r is too big or p_1 is too small” and terminate the algorithm; otherwise proceed to the next step.
2. [Pick roots and build z_{ij}] For $i = 1, \dots, s$, do
 For $j = 0, \dots, r$, do
 $k_{ij} \leftarrow j$
 $z_{ij} \leftarrow (m - k_{ij})^{-1} f(k_{ij}) \bmod p_i^{e_i}$.
 If $k_{ij} \equiv m \bmod p_i^{e_i}$ for some j , then set $k_{ij} \leftarrow r + 1$ and recalculate z_{ij} .
 Note: there will be at most one such j for each i .
3. [Build b_{ij}] For $i = 1, \dots, s$, calculate $(b_{i0}, \dots, b_{ir})^T$ from (1).
4. [Build b_i using CRT] For $j = 0, 1, \dots, r$, solve
 $b_j \equiv b_{1j} \bmod p_1^{e_1}$
 $b_j \equiv b_{2j} \bmod p_2^{e_2}$
 \vdots
 $b_j \equiv b_{sj} \bmod p_s^{e_s}$
 using CRT.
5. [Build $g(X)$] Define c_i as in (2) and set
 $g(X) \leftarrow c_d X^d + \dots + c_0$.
6. [Output and Terminate] Return $g(X)$ and $\{k_{ij}\}$ and terminate the algorithm.

5 Translations

Let us fix a polynomial $f(X) = a_d X^d + \dots + a_0 \in \mathbb{Z}[X]$ with $a_d \neq 0$. Note that for $\alpha \in \mathbb{Z}$, the roots of $f(X - \alpha) \in \mathbb{Z}[X]$ will just be the roots of f translated by α . However, the coefficients of $f(X - \alpha)$ will not (in general) be the coefficients of f . So $f(X - \alpha)$ has the same root properties as f , but perhaps differs in size. We now examine the effect of translation on the coefficients of f .

We define $T_f(U) = \{f(X - \alpha) \mid \alpha \in U\}$, where we will be interested in the cases $U = \mathbb{Z}, \mathbb{R}$. Also, fix $\omega = (\omega_0, \dots, \omega_d) \in \mathbb{R}^{d+1}$ and let

$$\|a_d X^d + \dots + a_0\|_{\omega, \infty} = \max_{0 \leq i \leq d} |\omega_i a_i|$$

$$\|a_d X^d + \dots + a_0\|_{\omega, k} = \left(\sum_{i=0}^d |\omega_i a_i|^k \right)^{\frac{1}{k}}$$

We will use the more convenient notation $\|\cdot\|_\infty$ and $\|\cdot\|_k$ and drop ω from the notation. Since polynomials with small coefficients tend to have small size, we will refer to $\|f\|_\infty$ as the *size* of f . For our fixed f and ω , we seek $h \in T_f(\mathbb{Z})$ with minimal size. The following proposition is the first step in finding such an h .

Proposition 1. Fix $f \in \mathbb{Z}[X]$ with $\deg(f) = d$, and $\omega \in \mathbb{R}^{d+1}$. Let $k \geq 1$ and take $g_k, h \in T_f(\mathbb{Z})$ with

$$\|g_k\|_k = \min_{p \in T_f(\mathbb{Z})} \|p\|_k$$

$$\|h\|_\infty = \min_{p \in T_f(\mathbb{Z})} \|p\|_\infty .$$

Then

$$\|g_k\|_\infty \leq (d+1)^{\frac{1}{k}} \|h\|_\infty .$$

Proof. Let $g_k = b_d X^d + \cdots + b_0$, and $h = c_d X^d + \cdots + c_0$. It is easy to see that $\|g_k\|_\infty \leq \|g_k\|_k$. Let $r_k = \|g_k\|_k$ and suppose that $|\omega_i c_i| < r_k / (d+1)^{\frac{1}{k}}$ for all i . Then

$$\|h\|_k^k = \sum_{i=0}^d |\omega_i c_i|^k < (d+1) \left(\frac{r_k}{(d+1)^{\frac{1}{k}}} \right)^k = r_k^k = \|g_k\|_k^k$$

which implies that $\|h\|_k < \|g_k\|_k$, a contradiction since $\|g_k\|_k$ is minimal in $T_f(\mathbb{Z})$. So there must be some i such that $|\omega_i c_i| \geq r_k / (d+1)^{\frac{1}{k}}$. But this means that $\|h\|_\infty \geq r_k / (d+1)^{\frac{1}{k}}$, which immediately implies $\|g_k\|_k \leq (d+1)^{\frac{1}{k}} \|h\|_\infty$.

Proposition 1 gives us the tool we need to find h .

Corollary 1. If $k > \frac{\ln(d+1)}{\ln(1+\|f\|_\infty^{-1})}$, then $\|g_k\|_\infty = \|h\|_\infty$.

Proof. For $k \geq 1$, Proposition 1 says that

$$0 \leq \|g_k\|_\infty - \|h\|_\infty \leq ((d+1)^{\frac{1}{k}} - 1) \|h\|_\infty .$$

But since $\|h\|_\infty \leq \|f\|_\infty$, we have

$$0 \leq \|g_k\|_\infty - \|h\|_\infty \leq ((d+1)^{\frac{1}{k}} - 1) \|f\|_\infty .$$

Now $((d+1)^{\frac{1}{k}} - 1) \|f\|_\infty \rightarrow 0$ as $k \rightarrow \infty$. But since $\|g_k\|_\infty - \|h\|_\infty$ is a nonnegative integer, as soon as $((d+1)^{\frac{1}{k}} - 1) \|f\|_\infty < 1$, we must have $\|g_k\|_\infty = \|h\|_\infty$.

Notice that although Corollary 1 gives us a way of finding h with minimal size in $T_f(\mathbb{Z})$ in theory, there is little hope of using this result in practice. In fact, $\ln x \approx x - 1$ when $x \approx 1$, so the denominator of the lower bound will be approximately $\|f\|_\infty^{-1}$, when $\|f\|_\infty$ is large. If we were to try to use Corollary 1, we would end up having to find the critical points of a degree kd polynomial, as we shall see shortly, which is clearly unreasonable when k is very large. With that said, Proposition 1 says that even for small k we can generate a polynomial with size equal to a rather small constant times $\|h\|_\infty$. With this in mind, let us now consider how we can find g_k as defined in Proposition 1. Observe that

$$\begin{aligned}
f(X - \alpha) &= \sum_{i=0}^d a_i (X - \alpha)^i \\
&= \sum_{i=0}^d a_i \sum_{j=0}^i \binom{i}{j} (-\alpha)^{i-j} X^j \\
&= \sum_{j=0}^d \left[\sum_{i=j}^d a_i \binom{i}{j} (-\alpha)^{i-j} \right] X^j \\
&= \sum_{j=0}^d p_j(\alpha) X^j,
\end{aligned}$$

where

$$p_j(\alpha) = \sum_{i=j}^d a_i \binom{i}{j} (-\alpha)^{i-j}.$$

Now define $m_k(\alpha) = \sum_{i=0}^d (\omega_i p_i(\alpha))^k$ and change variables to get $m_k(X) = \sum_{i=0}^d (\omega_i p_i(X))^k$. Let k be even. Then $m_k(X)$ is a polynomial of degree kd , with $m_k(X) \geq 0$ and $m_k(\alpha) = \|f(X - \alpha)\|_k^k$. Finding the value in \mathbb{Z} at which $m_k(X)$ achieves its absolute minimum is a straightforward task for small k .

We summarize this discussion with the following algorithm:

Algorithm 3. (Translation) Let $f \in \mathbb{Z}[X]$ be a polynomial of degree d , let k be a positive even integer and let $\omega \in \mathbb{R}^{d+1}$. This algorithm finds a polynomial $g_k \in \mathbb{Z}[X]$ and $\alpha_k \in \mathbb{Z}$, with $g_k(X) = f(X - \alpha_k)$ and $\|g_k\|_\infty$ less than or equal to $(d+1)^{\frac{1}{k}}$ times the size of a polynomial in $T_f(\mathbb{Z})$ of minimal size. In the process of computing g_k , the algorithm will compute the critical points of a polynomial of degree kd . If $k \geq \kappa := \left\lceil \frac{\ln(d+1)}{\ln(1+\|f\|_\infty)} \right\rceil$, then the algorithm will instead compute the critical points of a polynomial of degree κd , and g_k will have minimal size in $T_f(\mathbb{Z})$.

1. [Generate κ] Set $\kappa \leftarrow \left\lceil \frac{\ln(d+1)}{\ln(1+\|f\|_\infty)} \right\rceil$
 $\text{minimal?} \leftarrow \text{false}.$
2. [Is k too big?] If $k \geq \kappa$, then set $k \leftarrow \kappa$ if κ is even;
otherwise set $k \leftarrow \kappa + 1$
 $\text{minimal?} \leftarrow \text{true}.$
3. [Generate translate coefficients] For $j = 0, 1, \dots, d$, set $p_j(X) \leftarrow \sum_{i=j}^d a_i \binom{i}{j} (-X)^{i-j}.$
4. [Build $m_k(X)$] Set $m_k(X) \leftarrow (\omega_d p_d(X))^k + \dots + (\omega_0 p_0(X))^k.$
5. [Find critical numbers] Find $\alpha_{k1}, \dots, \alpha_{kl}$ such that $m'_k(\alpha_{ki}) = 0$ for all i .
6. [Identify α_k] Find $\alpha_k \in \{\lfloor \alpha_{ki} \rfloor, \lceil \alpha_{ki} \rceil\}_{i=1}^l$ such that $m_k(\alpha_k) \leq m_k(\lfloor \alpha_{ki} \rfloor), m_k(\lceil \alpha_{ki} \rceil)$ for all i .
7. [Build g_k] Set $g_k \leftarrow f(X - \alpha_k).$
8. [Output and Terminate] Return α_k and g_k . If $\text{minimal?} = \text{true}$, print "This polynomial has minimal size in $T_f(\mathbb{Z})$." In either case, terminate the algorithm.

6 Candidate NFS Polynomials

We can use Algorithms 1-3 to generate candidate NFS polynomials. More precisely, let us fix a positive integer N that we wish to factor and $d \geq 1$. Pick a suitable leading coefficient L , divisible by many small primes. We use Algorithm 1 to generate a polynomial f_1 of degree d with leading coefficient L and root m modulo N . We can then use Algorithm 2 to rotate f_1 by a polynomial of degree $r = d - 2$. This generates a polynomial f_2 with at least $d - 1$ roots modulo each element in some fixed set \mathcal{S} of small powers of distinct primes. Also, f_2 has leading coefficient L and root m modulo N . Finally, we use Algorithm 3 with a suitable choice for ω to produce a polynomial f_3 which has all the root properties as f_2 , with perhaps minimal size in $T_{f_2}(\mathbb{Z})$.

At this point, we have a candidate NFS polynomial with good root properties and small size. However, if this polynomial is not satisfactory for some reason, adjustments can be made to generate more polynomials. For example, we may generate many candidate NFS polynomials by varying i and L in Algorithm 1, \mathcal{S} in Algorithm 2, or ω in Algorithm 3. The following algorithm combines Algorithms 1-3 to produce candidate NFS polynomials:

Algorithm 4. (NFS candidate polynomial) Let $N \geq 1$ be a number that we wish to factor, L, d , and i be positive integers, $\mathcal{S} = \{p_1^{e_1}, \dots, p_s^{e_s}\}$ be a finite set of small powers of distinct primes, and $\omega \in \mathbb{R}^{d+1}$. This algorithm attempts to produce a candidate NFS polynomial with at least $d - 1$ roots modulo every $p_i^{e_i} \in \mathcal{S}$.

1. [Generate a base- m polynomial] Generate m and f_1 from Algorithm 1, using inputs i, d, L and N . If Algorithm 1 returns an error message, print the error message and terminate the algorithm.
2. [Rotate f_1] Generate f_2 and $\{k_{ij}\}$ from Algorithm 2, using inputs $f_1, d, m, N, \mathcal{S}$, and $r = d - 2$. If Algorithm 2 returns an error message, print the error message and terminate the algorithm.
3. [Translate f_2] Generate f_3 and α from Algorithm 3, using inputs f_2, d , and ω . If Algorithm 3 generates a message, print the message.
4. [Translate k_{ij}] For all i, j set:
 $k_{ij} \leftarrow k_{ij} + \alpha$.
5. [Output and Terminate] Return f_3 and $\{k_{ij}\}$ and terminate the algorithm.

7 Conclusion

One may wonder how “good” candidate NFS polynomials generated by Algorithm 4 will be. Let $f \in \mathbb{Z}[X]$ have degree d and $F \in \mathbb{Z}[X, Y]$ be the homogenization of f . One measure of “goodness” is

$$\alpha_B(F) = \sum_{p \leq B} \left(1 - q_p \frac{p}{p+1}\right) \frac{\ln p}{p-1}$$

where the sum is over all well-behaved primes (primes that do not divide the discriminant of f) less than or equal to some bound B , and q_p is the number of roots (regular and projective) of F modulo p , as defined in [2]. Heuristically and roughly speaking, we expect a typical value $F(x, y)$ to behave like a random integer of size $F(x, y) \cdot e^{\alpha_B(F)}$. So the more negative $\alpha_B(F)$ is, the “better” F should be. But clearly $\alpha_B(F)$ will be more negative whenever q_p is large for small primes p . Now $0 \leq q_p \leq d + 1$. However, by using Algorithm 4 we can force $q_p \geq d$ for each $p_i | L$ with $p_i^{e_i} \in \mathcal{S}$. If $\mathcal{S} = \{p_1^{e_1}, \dots, p_s^{e_s}\}$ with $p_1 < p_2 < \dots < p_s \leq B$, with $r \leq p_1 - 2$, then we will have a polynomial F which very likely has $\alpha_B(F) \ll 0$. Finally, by adjusting the coefficients after the CRT-step, or by using Algorithm 3, one hopefully has a suitable polynomial for factoring N using the Number Field Sieve. Future work will be devoted to identifying optimal parameters (i.e. L, \mathcal{S}, ω) for a given N .

Acknowledgements

The author wishes to thank Samuel S. Wagstaff, Jr. for valuable conversations concerning this work, as well as the anonymous referees for their useful comments on the paper.

References

1. A.K. Lenstra and H.W. Lenstra, Jr., editors. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993.
2. Brian Murphy. *Polynomial Selection for the Number Field Sieve Integer Factorisation Algorithm*. PhD thesis, Australian National University, July 1999.

On Class Group Computations Using the Number Field Sieve

Mark L. Bauer¹ and Safuat Hamdy²

¹ University of Waterloo
Centre for Applied Cryptographic Research
Waterloo, Ontario, N2L 3G1
`mbauer@math.uwaterloo.ca`

² University of Calgary
Department of Mathematics and Statistics
Calgary, Alberta, T2N 1N4
`hamdy@math.ucalgary.ca`

Abstract. The best practical algorithm for class group computations in imaginary quadratic number fields (such as group structure, class number, discrete logarithm computations) is a variant of the quadratic sieve factoring algorithm. Paradoxical as it sounds, the principles of the number field sieve, in a strict sense, could not be applied to number field computations, yet. In this article we give an indication of the obstructions.

In particular, we first present fundamental core elements of a number field sieve for number field computations of which it is absolutely unknown how to design them in a useful way. Finally, we show that the existence of a number field sieve for number field computations with a running time asymptotics similar to that of the genuine number field sieve likely implies the existence of an algorithm for elliptic curve related computational problems with subexponential running time.

Keywords: imaginary quadratic number fields, class groups, number field sieve, imaginary quadratic function fields, hyperelliptic curve discrete logarithm.

1 Introduction

The best practical algorithm for class group computations in quadratic number fields so far is a variant of the quadratic sieve algorithm. In the imaginary quadratic case such computations include the computation of class structures, class numbers, discrete logarithms, and Diffie-Hellman secrets; in the real quadratic case such computations include the computation of regulators, fundamental units, and principal ideal generators. In this article we focus on the imaginary quadratic case, though, some arguments may be generalized to the real quadratic case or even to the case of number fields of arbitrary degree.

We refer to the quadratic sieve algorithm for the imaginary quadratic case as by the IQ-MPQS. The IQ-MPQS has an asymptotic running time proportional

to $L_{|\Delta|}[\frac{1}{2}, c_1 + o(1)]$ for some positive constant c_1 (numerical evidence strongly suggests that $c_1 = 1$), where Δ is the discriminant.

1.1 Our Result

It is tempting to ask whether the number field sieve could not be used for number field computations as well. In fact, before the invention of the number field sieve the quadratic sieve was the best known algorithm to factor large integers, and the principles of the number field sieve could be profitably applied to many other computational problems which admit to algorithms of index-calculus type. However, paradoxical as it sounds, the number field sieve does not seem to work for number field computations.

In this article we give an indication of the obstructions in the imaginary quadratic case; we refer to the number field sieve in this case as by IQ-NFS. It must be clear, though, that if we ask for an IQ-NFS, then we mean to find an algorithm that is superior to the IQ-MPQS, i.e. having an asymptotic running time proportional to $L_{|\Delta|}[\frac{1}{3}, c_2 + o(1)]$ for some positive constant c_2 , or even $L_{|\Delta|}[\frac{1}{2}, c_3 + o(1)]$ for some positive constant c_3 non-negligibly smaller than 1.

By examining the connection between the number field computations and function field computations, we also show that an IQ-NFS with a running time proportional to $L_{|\Delta|}[\frac{1}{3}, c_4 + o(1)]$ for some positive constant c_4 could almost certainly be exploited to develop an algorithm for elliptic curve related computational problems.

We will conclude that if there exists an IQ-NFS, it will most likely not be superior to the IQ-MPQS, and if it did, its design would probably not follow that of the genuine NFS.

We must point out that this article is of somewhat speculative nature, and thus it should be understood as a starting point for further research.

1.2 Cryptographic Relevance

We outline now briefly the cryptographic relevance of our results. There is a family of cryptographic public-key schemes based on the intractability of some computational problems with class groups of imaginary quadratic number fields [8]; we call these cryptographic schemes IQ-schemes. Due to the sparseness of independent computational problems that admit to efficient cryptographic schemes, these public-key schemes were introduced as an alternative to existing schemes. More precisely: the security of the cryptographic schemes that are used in practice is based on the intractability of very few families of independent computational problems. Moreover, rigorous and unconditional proofs of the intractability of any of those computational problems are not known. This has repeatedly raised concerns about public-key cryptography. It is therefore advisable to have some well worked out and independent alternatives available. IQ-cryptography provides such an alternative; IQ-schemes are secure (using standard definitions and models of security) and efficient (in a practical sense).

The best known algorithm to solve these computational problems is, as already mentioned, the IQ-MPQS with the running time asymptotic $L[\frac{1}{2}]$. Traditional cryptographic schemes are based on the intractability of factoring integers or finite field computations, and the best known algorithms to solve these computational problems are variants of the number field sieve with the running time asymptotic $L[\frac{1}{3}]$. Thus, it seems that there is a complexity theoretic gap between IQ-related computational problems and factoring or finite field related problems.

Such a gap implies that, with increasing security level, the sizes of cryptographic parameters (such as RSA moduli, finite field size, discriminants etc.) and thus operands in a cryptographic operation (such as computing a signature) grow faster for traditional schemes than for IQ schemes. In spite of the more complex IQ-arithmetic it follows that IQ-cryptography eventually outperforms traditional cryptography. It is clear, though, that IQ-cryptography is eventually inferior to elliptic curve cryptography for the same reason. Yet, for the time being, IQ cryptography can be, in principle at least, considered as an efficient alternative.

However, the main motivation of IQ-cryptography is not its efficiency. We finally mention that due to the fact that the orders of class groups are in general not efficiently computable, IQ-cryptography has applications where elliptic curves do not work, see e.g. [5].

1.3 Notation

We use the following common notation

$$L_x[\varepsilon, c] = \exp\left(c(\log x)^\varepsilon (\log \log x)^{1-\varepsilon}\right),$$

while $L[\varepsilon]$ is the abbreviation for $L_x[\varepsilon, c]$ for some variable x and some positive constant c .

2 Constructive Obstructions

In this section we present some obstructions one encounters if one wants to design an IQ-NFS along the lines of the genuine NFS.

2.1 A Brief Review of the NFS Relation Generation

We begin with a brief review of the relevant details of the NFS relation generation for DL computations in finite fields, see [7, 12–15] as well as [2] for all details. Let p be a prime and let \mathbb{F}_p be a finite field. Then let d be a suitably chosen (small) integer, take $m = \lfloor p^{1/d} \rfloor$, and for $0 \leq i \leq d$ let a_i be the digits of the base- m expansion of p , i.e. let a_i be non-zero integers such that $p = \sum_{0 \leq i \leq d} a_i m^i$. Then let $f(X)$ be the polynomial with coefficients a_i of degree d . Suppose that $f(X)$ is irreducible over \mathbb{Z} , and let α denote a root of $f(X)$. Since

$$f(m) \equiv 0 \pmod{p} \tag{1}$$

the map defined by $\phi(\alpha) \mapsto m$, is a ring homomorphism from $\mathbb{Z}[\alpha]$ to \mathbb{F}_p . Formally one is looking at $\mathbb{Z}[\alpha]$ -integers of the form $\theta = a - b\alpha$, of which the norm is $N(a - b\alpha) = F(a, b)$, where $F(X, Y)$ is the homogenized bivariate polynomial that corresponds to $f(X)$. A $\mathbb{Z}[\alpha]$ -integer θ is understood to be smooth if its norm is smooth. The main task in the sieving stage is to find a set of coprime integers a and b such that θ and $\phi(\theta)$ are simultaneously smooth. That is, $F(a, b)$ and $a - bm$ have to be smooth simultaneously. This is done by taking $G(X, Y) = F(X, Y)(X - mY)$ and sieving the bivariate polynomial $G(X, Y)$.

Based on the bounds on X , Y and the coefficients of G , and assuming that the values of $G(X, Y)$ behave like random integers with respect to smoothness probability, one gets that the running time is proportional to $L_p\left[\frac{1}{3}, c_4 + o(1)\right]$, where $c_4 = (64/9)^{1/3}$; moreover, $d = \lfloor (3 \ln p / \ln \ln p)^{1/3} \rfloor$.

In order to get the favorable running time for the NFS the following items are crucial:

1. The degree d of f (and thus of G) tends uniformly to infinity as p tends to infinity.
2. The size of the coefficients of G are of order $p^{1/d}$.
3. There is an efficient way to select a polynomial and thus an extension of \mathbb{Q} .
4. There is an efficiently computable homomorphism from $\mathbb{Z}[\alpha]$ to \mathbb{F}_p .
5. The sieving is done in the two domains $\mathbb{Z}[\alpha]$ and \mathbb{F}_p simultaneously.

We will outline below that it is unknown how to achieve any of these items in the number field case.

2.2 A Brief Review of the IQ-MPQS Relation Generation

Before we proceed we shall briefly review the IQ-MPQS relation generation, see [9, 10] for details. Let \mathcal{O}_Δ denotes the quadratic order of discriminant Δ . The objective is to find a set \mathcal{R} of *relations* R_i of the form

$$R_i : \prod_{\mathfrak{p}_j \in \mathcal{FB}} \mathfrak{p}_j^{e_{i,j}} \sim \mathcal{O}_\Delta . \quad (2)$$

Here \mathcal{FB} is the *factor base*, a set of primitive prime \mathcal{O}_Δ -ideals of the form (p, b) where $p \leq B$ for some bound B . For each prime ideal $\mathfrak{p}_j = (p_j, b_j)$ of \mathcal{FB} , let $b_j \geq 0$; the prime ideal $\bar{\mathfrak{p}}_j = (p_j, -b_j)$ will be represented by \mathfrak{p}_j^{-1} .

In order to generate a relation, a \mathcal{FB} -smooth \mathcal{O}_Δ -ideal is constructed. Let \mathfrak{a} be this ideal with the representation (a, b) . The corresponding binary quadratic form is $A(X, Y) = aX^2 + bXY + cY^2$, where $c = (b^2 - \Delta)/4a$. Now, if there are coprime integers x and y such that $ax^2 + bxy + cy^2 = a'$, then there exists another binary quadratic form $A'(X, Y) = a'X^2 + b'XY + c'Y^2$, which is equivalent to A , and in fact, the corresponding \mathcal{O}_Δ -ideal $\mathfrak{a}' = (a', b')$ is equivalent to \mathfrak{a} . (The integer b' can be efficiently computed from the integers a, b, c, x , and y .) Therefore, $\mathfrak{a}\mathfrak{a}'^{-1} \sim \mathcal{O}_\Delta$, and if \mathfrak{a}' is \mathcal{FB} -smooth, this constitutes a relation. The prime ideal factorization of \mathfrak{a}' can be obtained from the prime factorization of a' , and from b' and Δ .

In order to find x and y such that $A(x, y) = a'$ is smooth, we sieve the quadratic polynomial $A(X, Y)$; for simplicity, fix $Y = 1$. Then the sieving is performed almost exactly as in the MPQS factoring algorithm. Likewise, the selection of polynomials is exactly as in the MPQS factoring algorithm, including the self initialization technique.

The sieving step of the IQ-MPQS is, in a remote sense, similar to its counterpart in the NFS. In both algorithms polynomials are sieved. Yet, none of the crucial properties above are satisfied. In particular, the degree of the sieving polynomials is fixed, no matter how large Δ is, the size of the coefficients of the quadratic polynomials are of the order of $|\Delta|^{1/2}$, and the sieving takes place only in one domain.

2.3 Towards an IQ-NFS

In this section we try to build an IQ-NFS on top of the IQ-MPQS. We proceed rather naively and follow along the lines of the genuine NFS.

Finding a Suitable Extension and a Homomorphism. First we try to find a suitable extension of \mathcal{O}_Δ . In the genuine NFS there was a natural way to find an irreducible polynomial over \mathbb{Z} : we took p and from it computed an integer m and coefficients of a polynomial $f(x)$ such that $f(m) \equiv 0 \pmod{p}$; in particular, $f(m) = p$, see above. Now, p was the characteristic of the finite field, which is a prime. However, in the number field case, the characteristic is always 0. So, it remains to be seen what to put in the place of p in the number field case.

Now, recall that the procedure in the genuine NFS to find a polynomial is not only natural because it is very simple and efficient. More important, we get the necessary ring-homomorphism from $\mathbb{Z}[\alpha]$ to \mathbb{F}_p , and this homomorphism is very efficient to compute. It is this very connection between the polynomial and the homomorphism that makes, for instance, the difference between the generalized number field sieve (with rather large polynomial coefficients) and the special number field sieve (with very small polynomial coefficients). If the coefficients in the GNFS could be chosen freely, then there would be no difference between the GNFS and the SNFS. However, it is not known how to find a suitable homomorphism for arbitrary polynomials, and therefore, the polynomial must be chosen as described above.

The same is certainly true in the number field case, where we have the additional problem of what to put in the place of p . We note, though, that in the number field case we are interested in a group-homomorphism instead of a ring-homomorphism. For instance, let $K = \mathbb{Q}(\sqrt{\Delta})$ and let \mathcal{O}_K be the maximal order of K ; likewise, let L be an extension of K and let \mathcal{O}_L be the maximal order of L .

What we are looking for is a group homomorphism ψ that maps \mathcal{O}_L to \mathcal{O}_K . Since a (basic) IQ-NFS algorithm would search for pairs $(\mathfrak{A}, \mathfrak{a})$, where \mathfrak{A} is an \mathcal{O}_L -ideal and \mathfrak{a} is an \mathcal{O}_K -ideal, ψ must satisfy the following properties:

1. if \mathfrak{A} is smooth (in a suitable sense), then $\psi(\mathfrak{A})$ is smooth;
2. $\psi(\mathfrak{A}) \sim \mathfrak{a}$;
3. ψ is efficiently computable.

In the face of the fact that \mathfrak{A} and \mathfrak{a} must be found *simultaneously* (by sieving a polynomial), property 2. in conjunction with property 3. appear to be the hardest to satisfy. In fact, it is unknown how one could do that.

Finding a Polynomial with Small Coefficients. Suppose for the moment that we have surmounted the obstructions from the previous subsection. It is tempting to ask what one would get out of the algorithm, i.e. what would be its asymptotic expected running time. Following the design of the genuine NFS, we proceed naively in the following way:

1. Choose a suitable integer d .
2. Choose an extension over K , i.e. choose a polynomial $f(X) \in \mathcal{O}_K[X]$ such that

$$f(X) = \alpha_d X^d + \alpha_{d-1} X^{d-1} + \cdots + \alpha_0, \quad (3)$$

where $\alpha_i = a_i + b_i \sqrt{\Delta}$. Let $|a_i|, |b_i| \leq B$, where B is a bound, e.g. $B = |\Delta|^{1/d}$ (recall that we just proceed naively as in the genuine NFS).

3. For the sieving we need a polynomial over \mathbb{Z} . In order to get such a polynomial $f_{\mathbb{Z}}(X)$ from $f(X)$, rewrite $f(X) = a(X) + b(X)\sqrt{\Delta}$ and let $\bar{f}(X) = a(X) - b(X)\sqrt{\Delta}$ be the conjugate polynomial. Now let $f_{\mathbb{Z}}(X) = f(X)\bar{f}(X) = a^2(X) - b^2(X)\Delta$. Note that since we are dealing with imaginary quadratic number fields, $\Delta < 0$ and thus $f_{\mathbb{Z}}(X) = a^2(X) + b^2(X)|\Delta|$. Now it becomes apparent that $f_{\mathbb{Z}}(X)$ has coefficients of the order of $|\Delta|^{1+2/d}$, which turns out to be too large in order to get the $L[\frac{1}{3}]$ running time asymptotics, see below.
4. Finally, let $F_{\mathbb{Z}}(X, Y)$ be the homogenized form of $f_{\mathbb{Z}}(X)$. (We presume that a smooth value for $F_{\mathbb{Z}}(X, Y)$ would in some way give rise to a smooth \mathcal{O}_L -ideal.) Let $A(X, Y)$ be the binary quadratic form that corresponds to an \mathcal{O}_K -ideal \mathfrak{a} , take $G(X, Y) = F_{\mathbb{Z}}(X, Y)A(X, Y)$ and sieve $G(X, Y)$ for pairs (x, y) such that $G(x, y)$ is smooth. Since the coefficients of $F_{\mathbb{Z}}$ are of size $O(|\Delta|B^2) = O(|\Delta|^{1+2/d})$ and the coefficients of A are of size $O(|\Delta|^{1/2})$, the coefficients of G are of size $O(|\Delta|^{3/2+2/d})$.

For the running time analysis we use the following principle from [2, Section 10]: Let $L(Z) = \exp(\sqrt{\ln Z \ln \ln Z})$. In a sequence of $L(Z)^{\sqrt{2}+o(1)}$ random integers uniformly chosen from the interval $[0, Z]$ $S = L(Z)^{1/\sqrt{2}+o(1)}$ of them will be S -smooth, and this is the optimal choice for S in order to maximize the yield.

We apply this principle to the sequence of integers that we get from $G(X, Y)$ for X and Y ranging over certain intervals; here we assume that the integers $G(X, Y)$ have the same properties as ordinary integers with respect to smoothness-probability (this constitutes, as usual, the major heuristic leap in the running time analysis).

We have $Z = |G(X, Y)|$, and since we sieve two-dimensionally, as in the genuine NFS, we have $|X|, |Y| \leq M$ where $M = L(Z)^{1/\sqrt{2}+o(1)}$. Now we are in the position to perform a running time analysis as in [2], see also [4, Section 6.2.3].

- If d is fixed as $|\Delta| \rightarrow \infty$, then we get an asymptotic running time proportional to $L_{|\Delta|}[\frac{1}{2}, 3 + \frac{4}{d} + o(1)]$.
- If $d \rightarrow \infty$ as $|\Delta| \rightarrow \infty$, then we get an asymptotic running time proportional to $L_{|\Delta|}[\frac{1}{2}, \sqrt{3} + o(1)]$.

This means that the IQ-NFS (designed as above) performs in any case much worse than the IQ-MPQS. Now, $f_{\mathbb{Z}}$ as chosen above may not be the optimal polynomial for L . One could, for example, use lattice reduction methods to get a polynomial $\overline{f_{\mathbb{Z}}}$ with smaller coefficients, see for example Algorithm POLRED in [3, Algorithm 4.4.1]. However, the coefficients of such a polynomial will not be arbitrary small, and even if $f_{\mathbb{Z}}$ had coefficients of order $O(|\Delta|^{1/2+2/d})$ (which constitutes a substantial improvement), then the asymptotic running times got merely down to $L_{|\Delta|}[\frac{1}{2}, 2 + \frac{4}{d} + o(1)]$ and $L_{|\Delta|}[\frac{1}{2}, \sqrt{2} + o(1)]$, which is still worse than the IQ-MPQS. We will elaborate the effectiveness of polynomial reduction algorithms applied to our problem in the full version of the paper. Finally, changing the polynomial also changes the basis for element and ideal representation in L , and thus, this changes the homomorphism; that might be a major problem.

In order to get the typical NFS asymptotic $L[\frac{1}{3}]$, the coefficients of $G(X, Y)$ must have order of magnitude $|\Delta|^{O(1/d)}$. Since the coefficients of $A(X, Y)$ usually have order of magnitude $|\Delta|^{1/2}$ we must alter the design of the IQ-NFS. Let $F_{\mathbb{Z},1}(X, Y)$ and $F_{\mathbb{Z},2}(X, Y)$ irreducible polynomials with the desired properties, let L_1 and L_2 be the corresponding extension fields, let $G(X, Y) = F_{\mathbb{Z},1}(X, Y)F_{\mathbb{Z},2}(X, Y)$, and let ψ_1 and ψ_2 be ideal-homomorphisms that map \mathcal{O}_{L_1} -ideals and \mathcal{O}_{L_2} -ideals to \mathcal{O}_K -ideals. Now we require that if a smooth \mathcal{O}_{L_1} -ideal \mathfrak{A}_1 and a smooth \mathcal{O}_{L_2} -ideal \mathfrak{A}_2 are found simultaneously by sieving $G(X, Y)$, then $\psi_1(\mathfrak{A}_1)$ and $\psi_2(\mathfrak{A}_2)$ are also smooth and $\psi_1(\mathfrak{A}_1) \sim \psi_2(\mathfrak{A}_2)$. Still, it remains to be seen how $F_{\mathbb{Z},1}$ and $F_{\mathbb{Z},2}$ as well as $\psi_1(\mathfrak{A}_1)$ and $\psi_2(\mathfrak{A}_2)$ are to be chosen.

Summary. The major stumbling blocks on the way towards an IQ-NFS are firstly to find suitable extensions of imaginary quadratic number fields, which provide suitable ideal-homomorphisms. It is unknown how to find such extensions. Secondly, by the nature of sieving algorithms, the extensions are to be represented as irreducible polynomials over \mathbb{Z} . It is unknown how to find suitable irreducible polynomials with sufficiently small coefficients. The first obstruction says that it is not known how to design core elements of the IQ-NFS, and the second obstruction says that even so it is still unknown how the IQ-NFS will be of any use.

3 Relative Obstructions

In this section we will attempt to provide a connection between the aforementioned problems and those that arise in the case of elliptic curves and hyperelliptic curves. We begin by stating the following definition.

Definition 1. *The Discrete Logarithm Problem in \mathcal{G} is: given $\gamma, \gamma' \in \mathcal{G}$, find the smallest $n \in \mathbb{Z}_{>0}$ such that $\gamma^n = \gamma'$ if such an integer exists.*

If the group under consideration corresponds to the points on an elliptic curve, we call this the *Elliptic Curve Discrete Logarithm Problem*, and abbreviate it by ECDLP. Analogously, if the groups corresponds to the Jacobian of a hyperelliptic curve, we call this the *Hyperelliptic Curve Discrete Logarithm Problem* and denote it by HCDLP. Since an elliptic curve is just a hyperelliptic curve of genus one, we note that the ECDLP is just a particular instance of the HCDLP.

The majority of this section will describe how to take an instance of the ECDLP and convert it to an instance of an HCDLP for a curve of higher genus. That is, we will prescribe a technique for constructing a cover of an elliptic curve by a hyperelliptic curve of larger genus that forces an inclusion from the elliptic curve into the Jacobian of the hyperelliptic curve. While this is apparently a well known result, we include some details since they appear to be lacking from the literature.

We conclude the section by discussing how the existence of this map relates to the overall complexity of solving the ECDLP. We will also discuss how finding an algorithm of lower subexponential complexity for the HCDLP seems intrinsically linked to solving the analogous problem for imaginary quadratic number fields.

3.1 Jacobians of Hyperelliptic (and Elliptic) Curves

In the remaining sections, let \mathbf{K} denote an arbitrary field. We will mostly be interested in the case when $\mathbf{K} = \mathbb{F}_q$, but the majority of what follows applies to arbitrary fields. For our purposes, it suffices to define a hyperelliptic curve of genus g to be a curve given by an equation of the following form:

$$C : y^2 + h(x)y = f(x)$$

where $h, f \in \mathbf{K}[x]$ are such that $\deg h \leq g$ and $\deg f = 2g + 1$ or $2g + 2$ with f monic. Furthermore, no element in $\overline{\mathbf{K}} \times \overline{\mathbf{K}}$ may simultaneously satisfy

$$y^2 + hy - f = 0 \quad , \quad 2y + h = 0 \quad , \quad h'y - f' = 0 \quad .$$

These last criteria force the curve to have a smooth affine model, which simply makes calculations more palatable (and the statement about the genus correct). Every hyperelliptic curve inherently admits such a model, so this by no means limits our discussion. Furthermore, if the characteristic of \mathbf{K} is not 2, we will always take $h = 0$ (this is possible by completing the square on the left hand side). A hyperelliptic curve of genus one is an elliptic curve. The *function field* of C is defined to be

$$\overline{\mathbf{K}}(C) \cong \overline{\mathbf{K}}(x)[y]/(y^2 + hy - f) \quad .$$

Each element of the function field can be thought of as a map from C to $\overline{\mathbf{K}} \cup \{\infty\}$ (otherwise denoted as $\mathbb{P}_{\overline{\mathbf{K}}}^1$).

We now give a brief overview of the Jacobian of a hyperelliptic curve. For more complete details, see the appendix in [11]. A *divisor* on C is a formal sum

of points $D = \sum m_P P$ where $m_P = 0$ for all but finitely many points of C . The degree of a divisor is $\deg D = \sum m_P$. The set of all divisors on C forms a group and is denoted $\text{Div}(C)$. The subset of all divisors of degree zero is a proper subgroup and is denoted $\text{Div}^0(C)$.

We wish to consider the quotient of $\text{Div}^0(C)$ by the following subgroup. For any function $\gamma \in \overline{\mathbf{K}}(C)$, we may associate a divisor to γ by $(\gamma) = \sum m_P P$ where $m_P = \text{ord}_P(\gamma)$ is the order of the zero or pole of γ at P . Such divisors are said to be *principal divisors*. The set of all principal divisors is denoted by $\mathcal{P}(C)$. $\mathcal{P}(C)$ is a subgroup of $\text{Div}^0(C)$ because every principal divisor has degree zero, although this is by no means obvious from the above definitions.

The group that we are interested in is called the *Picard group* of C (in fact, we are interested in the degree zero part of the Picard group, but we will abuse the language slightly). The group is defined to be

$$\text{Pic}^0(C) \cong \text{Div}^0(C)/\mathcal{P}(C) .$$

$\text{Pic}^0(C)$ contains all of the arithmetic information about the Jacobian of C that we need. If we have a tower of fields, $\mathbf{K} \subseteq L \subseteq \overline{\mathbf{K}}$, then $G_L = \text{Gal}(\overline{\mathbf{K}}, L)$ has a natural action on $\text{Pic}^0(C)$ induced by its action on $\mathcal{P}(C)$ and $\text{Div}^0(C)$ (which conveniently agree). The fixed group under this action is denoted by $\text{Pic}_L^0(C)$. Obviously, we will be most interested in the case when $L = \mathbf{K}$.

While the above construction of $\text{Pic}^0(C)$ is mathematically rigorous, it is somewhat lacking from a computational perspective. We will not cover the details of how to perform arithmetic, but instead refer the reader to the appendix in [11] again. The second computational problem that arises is how to represent elements in this group. For each element in $\text{Pic}^0(C)$, it is possible to associate to it a unique divisor in $\text{Div}^0(C)$. These unique divisors are called *reduced divisors*. The arithmetic and presentation in $\text{Pic}^0(C)$ are performed using these reduced divisors.

3.2 Including Elliptic Curves into Jacobians of Hyperelliptic Curves

We begin with a definition to facilitate in constructing the desired cover of our elliptic curve. Let p denote the characteristic of the field \mathbf{K} . For an integer n , we will write $n = n_1 \cdot p^{n_p}$ where n_1 is an integer that satisfies $\gcd(n_1, p) = 1$, and $n_p \geq 0$ (in characteristic zero, one takes $n_1 = n$, $n_p = 0$). Let $\mathfrak{p}_p(x)$ denote the Artin-Schreier character, i.e. $\mathfrak{p}_p(x) = x^p - x$. Define the polynomial

$$\mathfrak{C}_n(x) = \mathfrak{p}_p(x)^{\circ n_p} \circ x^{n_1} .$$

Theorem 1. *Let E be an elliptic curve given by*

$$E : y^2 + hy = f , \quad \deg f = 3 , \quad \deg h \leq 1 .$$

If in characteristic 2, $h(0) \neq 0$ or in characteristic different from 2, $f(0) \neq 0$, then there exists a hyperelliptic curve C_n of genus $\lfloor n + \frac{n-1}{2} \rfloor$ given by

$$C_n : y^2 + h(\mathfrak{C}_n(x))y = f(\mathfrak{C}_n(x)) ,$$

such that C_n is an n -to-1 cover of E .

The restrictions on $h(0)$ and $f(0)$ ensure that the resulting model for C_n is smooth. The smoothness of this model is a consequence of combining the definition of smoothness given above and noting that the polynomial given by $\mathfrak{C}_n(x) - \alpha$ for any $0 \neq \alpha \in \overline{\mathbf{K}}$ has no repeated roots. The genus follows trivially from the definition given above for a hyperelliptic curve.

Considering an elliptic curve over \mathbf{K} , it is possible to transform it into a curve of this form by using the substitution $x \mapsto x + \alpha$ for some $\alpha \in \mathbf{K}$ satisfying $h(\alpha) \neq 0$ in characteristic 2, or $f(\alpha) \neq 0$ otherwise. The only elliptic curve for which finding such an α is not possible is the curve

$$E : y^3 = x(x-1)(x-2)$$

defined over \mathbb{F}_3 . By extending the ground field, the above substitution could then be used. However, this curve is of no interest for the problem we wish to solve, so the above theorem applies to all cryptographically interesting elliptic curves.

The map from C_n to E is given as follows.

$$\Psi_n : C_n \rightarrow E$$

$$(\alpha, \beta) \mapsto (\mathfrak{C}_n(\alpha), \beta) ,$$

and the point(s) at infinity on C_n map to the unique point at infinity on E . This is clearly a well-defined algebraic map of curves, and for most points, there are precisely n distinct pre-images under Ψ since $\mathfrak{C}_n(x) - \alpha$ has degree n . Therefore, C_n is an n -to-1 cover of E via Ψ_n .

Given any two curves and a map between them, there is an induced map on the Jacobians of the two curves. We can use the map Ψ_n defined above to do precisely this. We proceed by constructing a map between the respective divisor class groups

$$\Psi_n^* : \text{Div}^0(E) \rightarrow \text{Div}^0(C_n) .$$

We define the map as follows. Let $P = (\alpha, \beta)$ be a finite point on E , and let α_i denote the n roots of $\mathfrak{C}_n(x) - \alpha$ (each with appropriate multiplicity). If n is odd, then

$$\Psi_n^* : P - P_\infty \rightarrow \left(\sum_{i=1}^n (\alpha_i, \beta) \right) - nP_\infty$$

where P_∞ represents the unique point at infinity on the two respective curves. If n is even,

$$\Psi_n^* : P - P_\infty \rightarrow \left(\sum_{i=1}^n (\alpha_i, \beta) \right) - \frac{n}{2} (P_{\infty_1} + P_{\infty_2})$$

where P_∞ is the unique point at infinity on E , and P_{∞_1} and P_{∞_2} are the two points at infinity on C_n . Since Ψ_n^* includes $\mathcal{P}(E)$ into $\mathcal{P}(C)$, it induces a map on the Picard groups, called the *conorm* map:

$$\text{Con}_{C_n/E} : \text{Pic}^0(E) \rightarrow \text{Pic}^0(C_n) .$$

It is precisely this map that we will use to translate an ECDLP into an HCDLP for a curve of higher genus.

Theorem 2. *If E and C_n are as specified in theorem 1, then the induced map*

$$\text{Con}_{C_n/E} : \text{Pic}_L^0(E) \rightarrow \text{Pic}_L^0(C_n)$$

is injective for all n and any $\mathbf{K} \subseteq L \subseteq \overline{\mathbf{K}}$.

By first proving the result over the algebraic closure, $\overline{\mathbf{K}}$, the theorem follows for all intermediary subfields by restriction. In our case, we are really only interested in the case of $L = \mathbf{K}$. For the case when n is odd, it is simple enough to show that the image under Ψ_n^* of a divisor on E of the form $P - P_\infty$, where P is any finite point, is a non-trivial reduced divisor in $\text{Div}^0(C_n)$. This is sufficient to conclude that the map is injective. If n is even, one proves the injectivity of the map by constructing a secondary hyperelliptic curve which is isomorphic to C_n , but has only one point at infinity.

Since this map is injective, it is clear that given an ECDLP for E , we can translate it into an HCDLP for C_n . This map is effective and quite easy to compute using \mathfrak{C}_n . If n is odd and we are using the standard representations for divisors on C_n , the map is given by

$$(\alpha, \beta) \mapsto \text{div}(\mathfrak{C}_n(x) - \alpha, \beta) .$$

3.3 Relating the Complexity of the ECDLP and HCDLP

Although C_n can be used to convert an ECDLP into an HCDLP, this does not necessarily help us solve the problem more efficiently. In fact, with the current algorithms for solving instances of HCDLP's, this amounts to taking a hard problem and making it harder. However, in this section, we consider the ramifications of the development of an algorithm to solve the HCDLP that is considerably more efficient than the algorithms that currently exist.

We start by noting that for a hyperelliptic curve of genus g over \mathbb{F}_q , the size of $\text{Pic}_{\mathbb{F}_q}^0(C)$ is roughly q^g .

Theorem 3. *If there exists an algorithm to solve the HCDLP with running time $L_{q^g}[\alpha, \beta + o(1)]$ with $\alpha < 1/2$ for $g \approx \log q$, as $q \rightarrow \infty$, then there exists an algorithm to solve the ECDLP in time $L_q[\alpha', \beta' + o(1)]$ with $\alpha' < 1$ and $\beta' \geq 0$ as $q \rightarrow \infty$.*

Proof. Given an elliptic curve E over \mathbb{F}_q , set $n = \lceil \frac{2}{3} \log q \rceil$. By using $\text{Con}_{C_n/E}$ and C_n which has genus $g \geq \log q$, we can solve the HCDLP in $\text{Pic}^0(C_n)$ in time $L_{q^g}[\alpha, \beta + o(1)]$. Letting γ be such that $g = \gamma \log q$, then we have

$$(\beta + o(1))(\log q^g)^\alpha (\log \log q^g)^{1-\alpha} = (\beta + o(1))(\log q)^{2\alpha} \gamma^\alpha (\log \gamma + 2 \log \log q)^{1-\alpha} .$$

Ignoring the coefficients for a moment, we may rewrite the right hand side as

$$(\log q)^{2\alpha+\epsilon} (\log \log q)^{1-(2\alpha+\epsilon)} \frac{(\log \gamma + 2 \log \log q)^{1-\alpha}}{(\log \log q)^{1-(2\alpha+\epsilon)} (\log q)^\epsilon}$$

for any $\epsilon > 0$. As $q \rightarrow \infty$, the fractional term tends to 0 (since $\gamma \rightarrow 1$), and hence we have that

$$(\log q^g)^\alpha (\log \log q^g)^{1-\alpha} \gg (\log q)^{2\alpha+\epsilon} (\log \log q)^{1-(2\alpha+\epsilon)} .$$

Therefore, an upperbound for the running time is given by $L_q[2\alpha + \epsilon, \beta']$, for any positive fixed $\epsilon > 0$, and any $\beta' \geq 0$ (although clearly both affect the constants involved in the big O-notation). If $\alpha < 1/2$, we can clearly choose $\epsilon > 0$ such that $2\alpha + \epsilon < 1$, which proves the desired result.

Theorem 4. *If there exists an algorithm to solve the HCDLP in time $L_{q^g}[\alpha, \beta + o(1)]$ with $\alpha = 1/2$ for $g \approx (\log q)^\delta$ and $\delta < 1$, as $q \rightarrow \infty$, then there exists an algorithm to solve the ECDLP in time $L_q[\alpha', \beta' + o(1)]$ with $\alpha' < 1$ and $\beta' \geq 0$ as $q \rightarrow \infty$.*

Proof. We proceed as above, but this time we choose $n = \lceil \frac{3}{2}(\log q)^\delta \rceil$. As before, we map to $\text{Pic}^0(C_n)$, where we can solve the HCDLP in time $L_{q^g}[\alpha, \beta + o(1)]$. Letting γ be such that $g = \gamma(\log q)^\delta$, then after substituting for g we note

$$\begin{aligned} (\beta + o(1)) (\log q^g)^\alpha (\log \log q^g)^{1-\alpha} &= (\beta + o(1)) (\log q)^{(1+\delta)\alpha} \gamma^\alpha \\ &\quad (\log \gamma + (1+\delta) \log \log q)^{1-\alpha} . \end{aligned}$$

Again ignoring the coefficients,

$$(\log q)^{(1+\delta)\alpha+\epsilon} (\log \log q)^{1-((1+\delta)\alpha+\epsilon)} \frac{(\log \gamma + (1+\delta) \log \log q)^{1-\alpha}}{(\log \log q)^{1-((1+\delta)\alpha+\epsilon)} (\log q)^\epsilon}$$

for any $\epsilon > 0$. We now note that as $q \rightarrow \infty$, the fractional term tends to 0. Hence we have that

$$(\log q^g)^\alpha (\log \log q^g)^{1-\alpha} \gg (\log q)^{(1+\delta)\alpha+\epsilon} (\log \log q)^{1-((1+\delta)\alpha+\epsilon)} .$$

This implies the running time is bounded above by $L_q[(1+\delta)\alpha + \epsilon, \beta' + o(1)]$ for any $\epsilon > 0$ and $\beta' \geq 0$. This time we note that if $\delta < 1$, and since $\alpha = 1/2$, we can choose ϵ so that $(1+\delta)\alpha + \epsilon < 1$.

It should not be construed that either of these two results imply the existence of a subexponential algorithm for the ECDLP. Examining them more deeply, they in fact suggest that our current index calculus approaches for solving the HCDLP are incapable of yielding algorithms with a running time in the range required for either theorem. To derive this conclusion we note that while the current algorithms for solving an instance of an HCDLP on a curve of genus g over \mathbb{F}_q utilize factor bases which are subexponential in terms of q^g , they are exponential in terms of q . Hence, using such algorithms to solve an instance of the ECDLP by embedding it in the Jacobian of a hyperelliptic curve can not result in a subexponential algorithm. This does not exclude the possibility of such techniques being effective in constructing an exponential algorithm which has a better run-time than Pollard-rho.

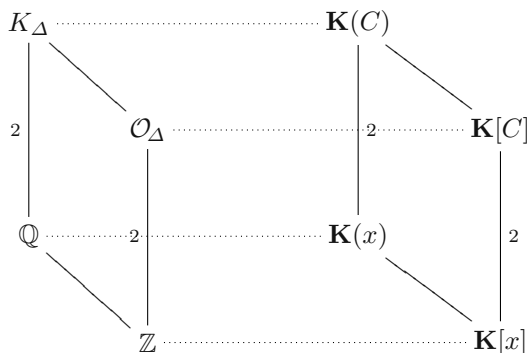
Using the same techniques as above, we can make the following somewhat perverse observation.

Theorem 5. *If there exists a subexponential algorithm for hyperelliptic curves of genus 2, then there exists a subexponential algorithm for elliptic curves.*

The proof follows as above using $n = 2$ (in fact, the previous theorem is also true with 2 replaced by any other fixed genus that may be written in the form $\lfloor n + \frac{n-1}{2} \rfloor$). It is important to note that the resulting algorithm would have worse overall complexity and the size of the elliptic curve for which the asymptotics would assert themselves is undoubtedly very large, but it would still be subexponential. The conundrum that arises from this observation is that the converse statement is not necessarily true.

3.4 The Analogue between HCDLP and IQDLP

While solving the HCDLP and IQDLP problems appear to be linked only superficially since they are both discrete logarithm problems, the connection between them runs much deeper. If we consider the original algorithm developed by [1] to solve the HCDLP in high genus hyperelliptic curves, it has the same fundamental structure as the IQ-MPQS. That is, they both find relations by searching for elements with smooth norms in certain quadratic extensions. If we restrict our attention to the case of imaginary hyperelliptic curves (when the degree of f is odd), then we have the following diagram can be used to demonstrate the connection.



In particular, solving the HCDLP in $\text{Pic}_{\mathbf{K}}^0(C)$ is equivalent to solving the same problem in the *ideal class group* of $\mathbf{K}[C]$. Fundamentally, both the HCDLP and class group computations in imaginary quadratic orders are equivalent to solving the discrete logarithm problem in ideal class groups of a quadratic extension. Indeed, if we consider the algorithm presented in [6], it can be considered to solve the problem in both situations. Although there are some subtle differences that arise in the analysis of the complexity, they do not serve to effect the the exponent α which has the greatest impact on the asymptotic run-time.

The problem that prevents the development of a suitable IQ-NFS is the same problem that prevents the development of a better algorithm to solve HCDLP's. Namely, how does one find an extension of a quadratic extension which yields suitable ideal-homomorphisms. Considering the strong analogy between the two

situations, it seems plausible that finding a solution in one of the two settings could easily be extended to the other.

4 Conclusion

We have presented some indication that the techniques of the number fields sieve may not be applicable to computations in imaginary quadratic number fields in a profitable way. In particular, it is unknown how to design fundamental core elements of an IQ-NFS algorithm, and even if this were known, it would not be clear whether or how such an algorithm could be useful (i.e. profitable). Moreover, we gave an outline how the existence of an IQ-NFS with the running time asymptotics $L\left[\frac{1}{3}\right]$ could conceivably be used to develop an algorithm to solve elliptic curve related computational problems with subexponential running time. It is worthwhile to point out that the analogy between these two settings is not restricted to algorithms of index-calculus type. It follows for example that if there is no subexponential algorithm to solve the ECDLP, then it is likely that $L_{|\Delta|}\left[\frac{1}{2}, c + o(1)\right]$ is the best achievable running time for the IQDLP.

As pointed out in the introduction, due to the somewhat speculative nature of this article, it is to be understood as a starting point for further research. For example, it would be interesting to establish rigorously the computational equivalence of the discrete logarithm problems for number field and function field class groups.

References

1. ADLEMAN, L. M., DEMARRAIS, J., AND HUANG, M.-D. A subexponential algorithm for discrete logarithms over hyperelliptic curves of large genus over $\text{GF}(q)$. *Theoretical Computer Science* 226, 1–2 (1999), 7–18.
2. BUHLER, J. P., LENSTRA, JR., H. W., AND POMERANCE, C. Factoring integers with the number field sieve. In *The development of the number field sieve*, A. K. Lenstra and H. W. Lenstra, Eds., no. 1554 in LNM. Springer-Verlag, 1993, pp. 50–94.
3. COHEN, H. *A Course in Computational Algebraic Number Theory*, vol. 138 of GTM. Springer-Verlag, 1995.
4. CRANDALL, R., AND POMERANCE, C. *Prime Numbers: A Computational Perspective*. Springer-Verlag, 2000.
5. DAMGÅRD, I., AND FUJISAKI, E. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology – ASIACRYPT 2002* (2002), Y. Zheng, Ed., vol. 2501 of LNCS, Springer-Verlag, pp. 125–142.
6. ENGE, A., AND GAUDRY, P. A. A general framework for subexponential discrete logarithm algorithms. *Acta Arithmetica* 102, 1 (2002), 83–103.
7. GORDON, D. M. Discrete logarithms in $gf(p)$ using the number field sieve. *SIAM Journal of Discrete Mathematics* 6, 1 (1993), 124–138.
8. HAMDY, S. IQ cryptography: A secure and efficient alternative. *Journal of Cryptology* (2003). Submitted.
9. JACOBSON, JR., M. J. Applying sieving to the computation of quadratic class groups. *Mathematics of Computation* 68, 226 (1999), 859–867.

10. JACOBSON, JR., M. J. *Subexponential Class Group Computation in Quadratic Orders*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 1999.
11. KOBLITZ, N. *Algebraic Aspects of Cryptography*, vol. 3 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 1998.
12. SCHIROKAUER, O. Discrete logarithms and local units. *Philosophical Transactions of the Royal Society of London, Series A*. 345, 1676 (1993), 409–423.
13. SCHIROKAUER, O. Using number fields to compute logarithms in finite fields. *Mathematics of Computation* 69, 231 (2000), 1267–1283.
14. SCHIROKAUER, O., WEBER, D., AND DENNY, T. Discrete logarithms: The effectiveness of the index calculus method. In *Algorithmic Number Theory, ANTS-II* (1996), H. Cohen, Ed., vol. 1122 of *LNCS*, Springer-Verlag, pp. 337–361.
15. WEBER, D. Computing discrete logarithms with the general number field sieve. In *Algorithmic Number Theory, ANTS-II* (1996), H. Cohen, Ed., vol. 1122 of *LNCS*, Springer-Verlag, pp. 391–403.

The Secret and Beauty of Ancient Chinese Padlocks

Hong-Sen Yan¹ and Hsing-Hui Huang²

¹ National Science and Technology Museum, Director General
720 Chiu-Ju 1st Road, Kaohsiung 807, Taiwan
hsyan@mail.ncku.edu.tw
<http://www.acmcf.org.tw>

² Department of Mechanical Engineering, Graduate student
National Cheng Kung University, 1 Ta-Hsueh Road, Tainan 701, Taiwan
sanly.huang@msa.hinet.net

Abstract. Most ancient Chinese padlocks are key-operated locks with splitting springs, and partially keyless letter-combination locks. They can be characterized based on the types of locks, the shapes of locks, the engravings of locks, the materials of locks, and the mechanisms of locks. Some locks and keys are not only very beautiful and artistic colorful, but also with various designs. As a result, a splitting spring padlock is an asymmetric key cryptosystem, and a combination padlock is a symmetric key cryptosystem.

1 Introduction

The development of locks arises psychologically from practical needs on safety for individuals, for groups, or for individuals within groups. Though with a long history, the related documents and perseverance of ancient Chinese locks are quite insufficient. And for their hardly noticeable nature in China, very few curio collectors set their eyes on locks, and very few scholars focused their study on locks [1, 2].

The history of Chinese locks is in close association with the materials, tools, and cultural background of a specific time. The development and applications of locks in the past reflected the technological, cultural, and economical situations of each period in the history. Ever since the late Eastern Han Dynasty, metal splitting spring padlocks had always been the most widely used locks by Chinese people. Though the shapes of ancient Chinese locks diversified, the inner structures have not changed much for the past two thousand years. And, Chinese locks faded gradually after the western pin-tumbler cylinder locks were introduced into the country in the 1940s.

This article addresses the beauty and the mechanisms of Chinese padlocks, and relates the opening systems of Chinese locks with cryptosystems.

2 Characteristics

Ancient Chinese locks are mechanical padlocks, mostly key-operated bronze locks with splitting springs and partially keyless letter-combination locks. The major features of ancient locks are the types of locks, the shapes of locks, the engravings of locks, the materials of locks, and the mechanisms of locks [3, 4].

2.1 Types of Locks

Ancient Chinese padlocks can be classified into the splitting spring locks and the letter-combination locks. A splitting spring padlock has to use a key for opening, and it has the types of broad locks and pattern locks. And, a letter-combination padlock has no keys for opening.

2.2 Shapes of Locks

Broad locks are kinds of horizontal positioned locks, Figure 1(a). The front side is of the shape of the character "凹", and mostly made of bronze. Pattern locks come in many different shapes, Figure 1(b). They can be roughly classified into the types of human figures, animals, musical instruments, letters, utensils, and others. Combination locks usually have three to seven wheels, Figure 1(c). They are of the horizontal round-pillar shape with several tunable wheels of the same size set in array on the central axis of the pillar body. Each wheel has the same amount of carved letters.



(a) Broad locks

(b) Pattern locks

(c) Combination locks

Fig. 1. Types and shapes of Chinese locks

2.3 Engraving of Locks

Engraving on the body surface of Chinese locks can be classified into two types: the etching and the engraving. Patterns commonly employed are lucky objects, human figures, Chinese characters, landscapes, flowers, plants, and others. All these revealed hidden handicraft skills and great beauty in an object of such utility.

2.4 Materials of Locks

According to the development of various materials in various periods, ancient Chinese locks were made of wood, bronze, brass, red bronze, Cupro nickel, iron, silver, gold, steel, aluminum, and nickel. The early broad locks found were mostly made of bronze; later the brass was the most popular, followed by iron.

3 Mechanisms

A splitting spring padlock consists of a lock-body, a sliding bolt, and a key, Figure 2(a). The lock-body provides a keyhole for the key to insert and the supporting guide for the sliding bolt to move. The sliding bolt has a shackle for hanging the lock and a stem for bonding one end of the splitting springs. The key is designed corresponding to the configuration of the splitting springs, and the location and shape of the keyhole. When it is locked, the sliding bolt is trapped by the opening springs against the inner wall of the lock-body. For opening, the key is inserted and its head squeezes the opening springs so that the sliding bolt can be separated from the lock-body.

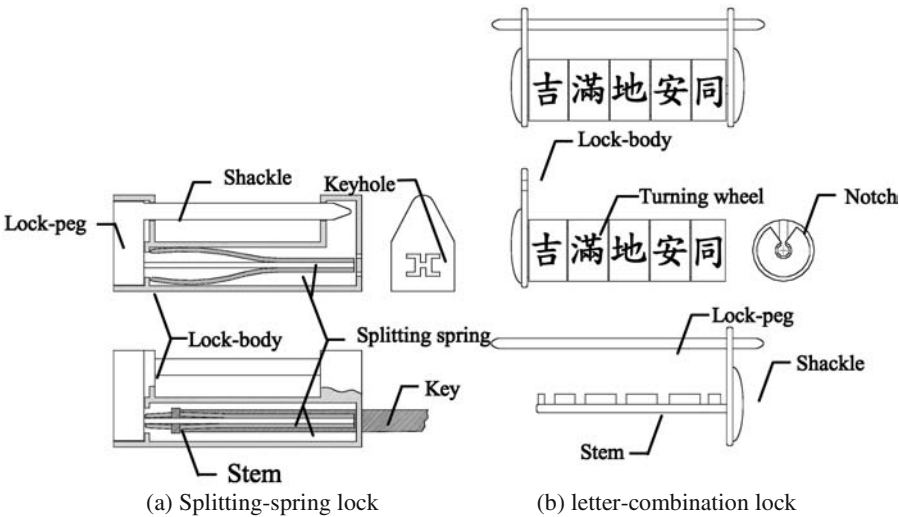


Fig. 2. Mechanisms of Chinese locks

A combination padlock comprises of the lock-body, rotating wheels, and the sliding bolt with a shackle and a stem, Figure 2(b). The lock-body contains an end plate and an axis with rotating wheels for guiding the movement of the sliding bolt. The sliding bolt also has an end plate for bonding both the shackle to hang the lock and the stem with several convex (凸)-shaped blocks. Every rotating wheel is of the same size. Usually four letters are engraved on the surface. And, there is a concave (凹)-shaped chute that corresponds with each convex-shaped block on the stem. When unlocking the lock, one has to rotate the letters on each wheel into the correct order and position. When all the concave-shaped chutes face upward, a channel is formed that allows the stem with convex-shaped blocks to slide apart from the lock-body. The lock is then opened.

4 Cryptosystems

A Chinese splitting spring padlock has the following features:

1. The opening key is designed to have the right shape of keyhead to be inserted through the designed shape of the keyhole and to squeeze the designed configuration of the splitting springs to open the lock.
2. It does not need the opening key to fasten the lock.
3. In general, a key is designed for a specific lock. However, sometimes a key can open more than one lock.

Therefore, a splitting spring padlock is an asymmetric key cryptosystem.

Furthermore, a Chinese combination padlock is a symmetric key cryptosystem. When the letters (ciphers) of all wheels are rotated into the right sequence, it is unlocked; otherwise, it is locked.

4.1 Basic Components

The mechanism of an ancient Chinese padlock has three basic components•a fastening device, an opening device, and an obstacle.

Chinese padlocks usually use a sliding or rotary bolt as the fasten component. Most Chinese padlocks used splitting springs as the obstacle to discriminate and obstruct the wrong opening devices. In general, there are two types of obstacles. On is the fix obstacle, such as the special keyholes or keyways to prevent the invasion of foreign keys to open (decrypt) the locks. And, the other can be moved by the inserting keys to strengthen the encryption of the locks, and also to screen the wrong keys that break the first layer of security - keyholes and keyways. The opening device is used to overcome the obstacle component and can be a key or a secret code.

4.2 Encrypt and Decrypt

Although the coding of locks always refers to the matching design of locks and keys, it can also fit on the operation of locks. Figure 3 shows the relationship of the fastening device, the opening device, and the obstacle component that construct the operation of a lock. The opening of a Chinese lock can be taken as the key of a splitting spring padlock or the cipher of a combination padlock. When fastening the lock, the opening device should be discriminated by the obstacle to ensure the validity and encrypt the lock. If the opening device is wrong, the correct one should be renewed for preceding the encoding of the lock. Once the obstacle is overcome, the fastening device is released and the lock is in the fastening condition.

5 Conclusions

Although locks have been used around our daily lives in the past thousands of years, the development and characteristics of ancient Chinese locks not only have been

almost unknown to the world but also have not been fully investigated. This paper presents the secret and beauty of ancient Chinese padlocks based on authors' study and collection in the past years [5]. Ancient Chinese locks are mostly key-operated bronze padlocks with splitting springs and partially letter-combination padlocks. Chinese padlocks can be characterized based on the mechanism of locks, the shape of locks, the type of keyways, the shape of keyways, the type of keys, the shape of key-heads, the insertion of keys, and the materials of locks. A splitting spring padlock is an asymmetric key cryptosystem, since it has to use a key for opening and it does not need the opening key to close the lock. A Chinese combination padlock is a symmetric key cryptosystem. When the letters of all wheels are rotated into the right positions, it is unlocked; otherwise, it is locked. It is hope that this article will induce further research interest to relate ancient Chinese locks and modern cryptosystems.

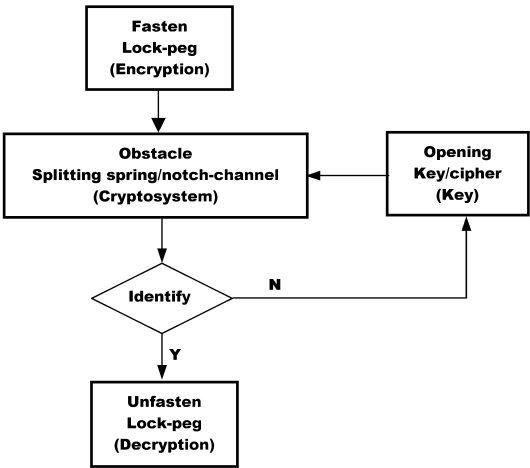


Fig. 3. Cryptosystem of Chinese locks

References

1. Joseph, N., Science and Civilization in China, Vol. IV, Cambridge University Press (1965).
2. Yan, H.S., The Beauty of Ancient Chinese Locks, 2nd edition, Ancient Chinese Machinery Cultural Foundation, Tainan, TAIWAN, ISBN 957-28702-0-X (2003).
3. Yan, H. S., "On the characteristics of ancient Chinese locks," Proceedings of the First China-Japan International Conference on History of Mechanical Technology, Beijing (1998) 215-220.
4. Yan, H.S. and Huang, H.H., "On the spring configurations of ancient Chinese locks," Proceedings of International Symposium on History of Machines and Mechanisms HMM2000, Cassino, Italy (2000) 87-92.
5. Yan, H.S., March 2003, "On the collection and research of ancient Chinese locks," Technology Museum Review, Vol.7, No.1, National Science and Technology Museum, Kaohsiung, TAIWAN (2003) 4-6.

A Traceable Block Cipher

Olivier Billet and Henri Gilbert

France Télécom R&D
38-40, rue du Général Leclerc
92794 Issy les Moulineaux Cedex 9 - France
{olivier.billet,henri.gilbert}@francetelecom.com

Abstract. In this paper¹ we propose a new symmetric block cipher with the following paradoxical traceability properties: it is computationally easy to derive many equivalent secret keys providing distinct descriptions of the same instance of the block cipher. But it is computationally difficult, given one or even up to k equivalent keys, to recover the so called meta-key from which they were derived, or to find any additional equivalent key, or more generally to forge any new untraceable description of the same instance of the block cipher. Therefore, if each legitimate user of a digital content distribution system based on encrypted information broadcast (e.g. scrambled pay TV, distribution over the Internet of multimedia content, etc.) is provided with one of the equivalent keys, he can use this personal key to decrypt the content. But it is conjectured infeasible for coalitions of up to k traitors to mix their legitimate personal keys into untraceable keys they might redistribute anonymously to pirate decoders. Thus, the proposed block cipher inherently provides an efficient traitor tracing scheme [4]. The new algorithm can be described as an iterative block cipher belonging to the class of multivariate schemes. It has advantages in terms of performance over existing traitor tracing schemes and furthermore, it allows to restrict overheads to one single block (*i.e.* typically 80 to 160 bits) per encrypted content payload. Its strength relies upon the difficulty of the “Isomorphism of Polynomials” problem [17], which has been extensively investigated over the past years. An initial security analysis is supplied.

Keywords: traitor tracing, block ciphers, Matsumoto-Imai, multivariate cryptology, symmetric cryptology, collusion resistance.

1 Introduction

One of the most employed digital content distribution methods consists in broadcasting encrypted information. Applications include pay TV systems, server-based services for the distribution of pre-encrypted music, videos, documents or programs over the Internet, distribution of digital media such as CDs or DVDs, and more generally, conditional access systems. In content distribution systems broadcasting encrypted information, each user is equipped with a “decryption

¹ This paper was submitted to the Asiacrypt 2003 conference.

box” which may be a smart card combined with an unscrambling device as in several existing pay TV systems, or even of software on a personal computer. The decryption box of each legitimate user is provided with a decryption key, allowing him to recover the plaintext content from the broadcast information during some validity period or for a given subset of the content. The delivery and update of decryption keys may be performed using various key distribution methods and is generally subject to the payment of subscriptions, digital right management licenses, etc.

The following security problem arises in this setting: if any legitimate user manages to recover the decryption key contained in his decryption box or to duplicate the keyed decryption software, then he can redistribute it to illegitimate users, allowing them to get the plain content as the legitimate users, without having to pay any subscription, digital right management license, etc. This quite often represents a much more serious threat than the redistribution of the plaintext content, which is so far not considered very practical in contexts like pay-TV. The use of tamper resistant devices (e.g. smart cards) to store decryption keys and associated algorithm(s) obviously helps protecting these systems, but can hardly be considered a sufficient countermeasure to entirely prevent this kind of attacks. Over the past years, more and more sophisticated attacks against tamper resistant devices have emerged—e.g. side-channel attacks, see for instance [12]. Because attacking a single decryption box may lead to massive fraud, attackers can afford using sophisticated and expensive attacks, so that countermeasures proposed in other contexts will often be ineffective for encrypted content broadcast systems.

Traitor tracing provides a natural countermeasure to prevent the decryption key redistribution threat described above. The concept of traitor tracing scheme was first introduced by B. Chor, A. Fiat and M. Naor in the seminal paper [4] and we use as far as possible the same terminology to describe the proposed scheme. In traitor tracing schemes, each legitimate user is provided with a unique personal decryption key which unambiguously identifies him, while enabling him to decrypt the broadcast information. The system must accommodate a large number N of users and it must be infeasible for any coalition of up to k legitimate users to mix their personal keys into a new untraceable description of the decryption key. Most of the traitor tracing schemes proposed so far, e.g. those described in [4], [14] and [18] are combinatorial in nature. Each legitimate user is provided with several base keys, which together form his personal key and the broadcast information contains large overheads of encrypted values under some of the base keys, allowing legitimate users to recover a content decryption key. A non-combinatorial alternative, namely a public key encryption scheme in which there is one public encryption key but many private decryption keys, was proposed by D. Boneh and M. Franklin in [3]. It has the advantage to avoid large overheads and to have very small decryption keys. However, the performance of this scheme is extremely sensitive to the maximum number k of tolerated colluding traitors, since the data expansion factor of the public key encryption is proportional to k .

The approach developed in this paper is non combinatorial in nature and has stronger connection with the one developed in [3] than with combinatorial schemes, up to the essential difference that we construct an untraceable symmetric cipher rather than an untraceable asymmetric cipher. The proposed cipher has the paradoxical property that many equivalent secret keys (used for decryption purposes) can be generated, while it is conjectured to be computationally impossible, given at most k equivalent secret keys, either to forge another untraceable equivalent secret key or to reconstruct the “meta key” from which the original equivalent secret keys were derived. More precisely, the knowledge of the meta key allows to efficiently determine at least one of the equivalent secret keys used to forge the new description.

The proposed construction can be described as an iterative block cipher. Its strength relies upon the intractability of the “Isomorphism of Polynomials,” a problem which has been extensively investigated over the past years [2,11,17] and which conjectured intractability has not been directly affected by recent advances in the cryptanalysis of multivariate schemes like HFE [9,10]. One of the advantages of the proposed scheme is to avoid generated overhead compared to the combinatorial approach taken in [3] where the data expansion is proportional to k . Another advantage is the intrinsic structure which is rather close to the one of usual block ciphers, so that the performance of the cipher in encryption/decryption modes is better than for existing traitors tracing schemes. Also the proposed scheme is much less sensitive to the maximum number k of traitors tolerated in a coalition, or to the maximum number of users N in the system. On the negative side, one should mention that the tracing procedures described in this paper require the knowledge of the description of the decryption function owned by a pirate. Thus no “black box” tracing procedure limiting interaction with the pirate decoder to “oracle queries” is provided. Another limitation of the proposed algorithm is that as usual in symmetric cryptography, no provable reduction to the difficulty of a well studied mathematical problem (e.g. the isomorphism of polynomial problem) could be found. Thus, the security analysis we supply can only achieve the next desirable goal, *i.e.* investigate various attack strategies and make sure that identified attacks are thwarted. Because of the higher requirements on a traceable cipher, risks are obviously much higher than for usual symmetric ciphers.

This paper is organized as follows. In Section 2, we describe the requirements on a symmetric cipher with an associated non-combinatorial traitor tracing scheme. In Section 3, we describe the proposed iterative block cipher construction and the associated traitor tracing scheme. Section 4 provides an initial security analysis. Section 5 addresses performance issues and provides an example instance of the proposed algorithm with explicit practical parameter values, in order to stimulate improved cryptanalysis. Section 6 concludes the paper.

2 Traceable Block Ciphers: Requirements and Operation

Let us denote by F_K , $K \in \mathbf{K}$ a symmetric block cipher of block size l , *i.e.* a key-dependent function from the set $\{0, 1\}^l$ of l -bit input values to itself. As will

be seen in the sequel, it is not required that F_K be easy to invert. It is not even an absolute requirement that the function F_K be one to one, although the block ciphers proposed in this paper are actually one to one and can be inverted: in practice they are operated in the forward direction alone, except in some traitor tracing procedures.

A traitor tracing scheme for N users associated with a traceable symmetric block cipher F_K consists of the following components:

- **A user initialization scheme** deriving users' secret keys $(K_j)_{j=1,\dots,N}$ from a meta key $K \in \mathbf{K}$. All user secret keys K_j must be distinct (though equivalent) descriptions F_{K_j} of the meta function F_K . Each description F_{K_j} must allow to efficiently compute F_K in the forward direction.
- **Encryption and decryption processes**, respectively used by the operator of the broadcast distribution system to encrypt some digital content using F_K , and by the legitimate user j to decrypt this content using his recovery key K_j through the associated description F_{K_j} of F_K . As explained in [4], the structure of the broadcast information typically consists of pairs (EB_i, CB_i) of an overhead information named "enabling block" and an encrypted content block named "cipher block." The enabling block is used to generate a symmetric key, hereafter called "control word," to decrypt the cipher block via an additional symmetric scheme S , like for instance AES or one-time pad. As said before, F_K needs not to be invertible: it is used in the forward direction in both the encryption and decryption processes.
- **A tracing procedure** allowing the owner of the meta key, when provided with any pirate description of the decryption function forged by any coalition of up to k traitors, to trace at least one traitor of the coalition.

In this setting, the meta key's holder creates cipher blocks CB_i from blocks of plain text content B_i using an additional symmetric scheme S and enabling blocks EB_i (produced for instance by a pseudo-random generator) via the formula $CB_i := S_{CW_i}(B_i)$, where the control words CW_i are derived from the enabling blocks using the traceable block cipher $CW_i := F_K(EB_i)$.

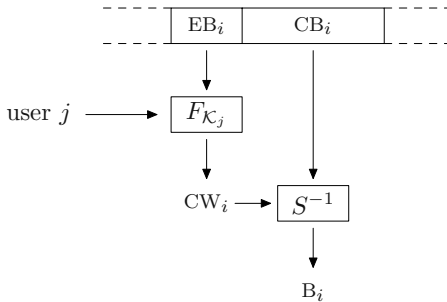


Fig. 1. Scheme's Architecture.

The operations performed by legitimate users to decrypt these content blocks are summarized in Fig. 1. User j first derives the control word CW_i from the enabling block EB_i via his description F_{K_j} of the meta function: $CW_i = F_{K_j}(EB_i)$. Then he uses the control word CW_i to decrypt the cipher block CB_i via the additional symmetric scheme S and recovers associated block(s) of plain content $B_i = S_{CW_i}^{-1}(CB_i)$. For instance, $B_i = CB_i \oplus CW_i$ when S is the one time pad algorithm, and $B_i = \text{AES}_{CW_i}^{-1}(CB_i)$ when S is the AES. In this context, control words must be frequently generated to prevent attacks by redistribution of these control words to pirate decryption boxes from being much easier than the redistribution of plaintext. This is difficult to achieve with existing combinatorial traitor tracing schemes due to the large data expansion incurred by such schemes. Another consequence is that the throughput (bit/s) of the F_K block cipher must be as close as possible to the throughput of classical block ciphers such as AES and much larger than the one of asymmetric ciphers such as RSA. An additional requirement for systems where K needs to be updated frequently, e.g. to manage dynamic modifications of lists of subscribers, is that each description F_{K_j} be reasonably short for the distribution via any symmetric encryption or key distribution algorithm to be practical.

In order for the content distribution system to resist attacks against the decryption scheme, the descriptions F_{K_j} must satisfy the usual security requirements of a block cipher. This implies that given any set of F_{K_j} input/output pairs with known, chosen or even adaptively chosen input values an adversary could obtain, it must be computationally infeasible for this adversary to predict any additional F_{K_j} input/output pair with a non negligible success probability. In particular input/outputs pairs must not reveal K_j or any other equivalent description of F_K .

The last and most demanding requirement is the existence of an efficient traitor tracing procedure for the owner of the meta key K . Our definition of a traitor tracing scheme follows the one proposed in the seminal paper [4]. We do not require the traitor tracing scheme to be black box (*i.e.* to be operable using say only inputs EB_i and outputs CW_i of the key distribution function). We restrict ourselves to traitor tracing scenarios where an authority is able to access the description of the description of F_K contained in the pirate decryption box. Note that it does not seem unrealistic to assume that decryption boxes of pirate users can be tampered by an authority, taking into account the fact that traitor tracing is only needed if the decryption boxes of legitimate users can be tampered. Traitor tracing requirements can be informally stated as follows. Attacks by any coalition of up to k traitors should be traceable, that is k traitors able to access their individual descriptions F_{K_j} should not be computationally able to forge any additional description F' from their k equivalent descriptions F_{K_j} without revealing at least one of their K_j —and thus the identity j of one of the traitors. We further require that the probability for the tracing procedure applied to any k -traitors coalition to either output no suspected traitor (non detection) or to output the identity j of an innocent user (false alarms) be negligible.

3 Description of the Traceable Scheme

Among the requirements identified in the former Section, the most demanding one is not the existence of many equivalent descriptions of the symmetric function F_K —this is frequent in symmetric cryptography, see for instance [1]—but the property that the provision to a user of one of these numerous representations F_{K_j} should not disclose information allowing him to construct any other representation of F_K unrelated to K_j . In other words, the meta key K must act as a kind of trapdoor allowing to perform other operations than those allowed by the descriptions F_{K_j} of F_K . Thus, even in the symmetric setting considered in this paper, public key cryptography properties are required and generic block ciphers will not be usable like in the case of combinatorial traitor tracing schemes. However we would like to keep performance advantages of symmetric cryptography since generation of control words at high rate is necessary for the security of the system.

Multivariate cryptography appears to be a natural candidate to meet these requirements. As a matter of fact, features of this recently developed family of algorithms are to many extents intermediate between those of public key algorithms (e.g. trapdoors) and those of secret key algorithms. Many of them can be described as iterative ciphers resulting of the composition of several rounds, and their complexity is substantially lower than the one of usual public key ciphers and not much higher than the one of usual block ciphers. Typical examples of multivariate algorithms are C^* proposed by T. Matsumoto and H. Imai in [13], SFLASHv2 (one of the Nessie finalists [19]), and HFE [16]. All the schemes mentioned above rely on the intractability of the so-called “Isomorphism of Polynomials” problem for the *secret key recovery*. See [7] for more information about known attacks against this problem. The C^* scheme was attacked by Patarin in [15] and Dobbertin independently, but these attacks do not allow to recover the secret key and thus to break the underlying IP problem. An attack allowing to solve the IP problem underlying some instances of HFE, using so-called re-linearization techniques was published by Kipnis and Shamir in 1999 [11], and appears to be also applicable to the IP problem underlying some instances of the basic (quadratic) version of C^* . More recently, enhanced decryption or signature forgery attacks against HFE and more generally various multivariate cryptosystems have been proposed [8,6,9,10]. But none of these recent attacks allows to recover the secret key and to break the underlying IP problem. Thus in summary, as far as we know, the best known attacks against the IP problem underlying multivariate schemes are those described in [7,11].

3.1 Building Blocks

Let us briefly recall the basic quadratic C^* from which the building block of our scheme is directly derived by generalizing it to monomials of higher degree. It involves the following elements:

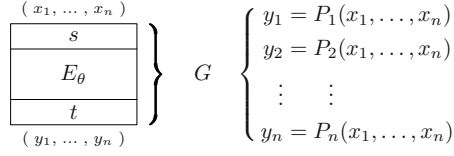


Fig. 2. An extended C^* building block.

- A finite field $\mathbb{K} = \mathbb{F}_q$ of size q .
- An extension \mathbb{L} over \mathbb{K} of degree n , with a defining primitive polynomial $P(X)$ of degree n such that $\mathbb{L} = \mathbb{K}[X]/(P(X))$. We will represent elements of \mathbb{L} as n -tuples (a_0, \dots, a_n) of \mathbb{K} through the usual identification function $\varphi : (a_0, \dots, a_n) \mapsto \sum_{i=0}^n a_i X^i \pmod{P(X)}$.
- A private key made of two linear one to one mappings s and t from \mathbb{K}^n to itself and an integer θ such that $q^\theta + 1$ be prime to $q^n - 1$.
- A public key $G = t \circ \varphi^{-1} \circ E_\theta \circ \varphi \circ s$, published as a system of n multivariate polynomials in n variables, where E_θ is a monomial function defined to be $\mathbb{L} \rightarrow \mathbb{L}$, $a \mapsto a^{1+q^\theta}$. Assuming the trapdoor (s, t) unknown, function G was believed to be one-way, but J. Patarin showed in [15] that it can be computationally inverted. However, one-wayness is not needed in our scheme.

The actual building blocks of our construction are higher degree variants of C^* obtained by considering a more generic—but still monomial—function E , namely $E_\Theta : \mathbb{L} \rightarrow \mathbb{L}$, $a \mapsto a^{1+q^{\theta_1}+\dots+q^{\theta_{d-1}}}$ where d is a fixed integer and Θ is a $(d-1)$ -tuple $(\theta_1, \dots, \theta_{d-1})$ such that $q^n - 1$ be prime to $1 + q^{\theta_1} + \dots + q^{\theta_{d-1}}$, hereafter called the degree of the building block G . Indeed, G can be described as a system of n multivariate polynomial equations as suggested in Fig. 2, and the polynomials P_i involved have total degree d . For instance, in the special case where $d = 3$, G can be described as $(i = 1, \dots, n)$:

$$y_i = \sum_{0 \leq j, k, l \leq n-1} \alpha_{i,j,k,l} x_j x_k x_l + \sum_{0 \leq j, k \leq n-1} \beta_{i,j,k} x_j x_k + \sum_{0 \leq j \leq n-1} \gamma_{i,j} x_j \quad (1)$$

The basic idea underlying the proposed traitor tracing scheme is to use several of those extended C^* instances as building blocks for our construction and to take opportunity of the commutativity of the various monomial functions E_θ involved—that is $E_{\theta_1} \circ E_{\theta_2} = E_{\theta_2} \circ E_{\theta_1}$ for all θ_1, θ_2 .

3.2 Meta-key, Users' Keys

Let us keep the notation of the previous Section. Moreover, let r be the number of building blocks. The meta secret key \mathcal{K} is defined as the set of two one to one linear mappings s and t from \mathbb{K}^n to itself, and a collection of r $(d-1)$ -tuples Θ_i such that all the values $1 + q^{\theta_{1,i}} + \dots + q^{\theta_{d-1,i}}$ for $i = 1, \dots, r$ be distinct. Then the function $F_{\mathcal{K}}$ is defined as $F_{\mathcal{K}} = s \circ E_{\Theta_r} \circ \dots \circ E_{\Theta_2} \circ E_{\Theta_1} \circ t$.

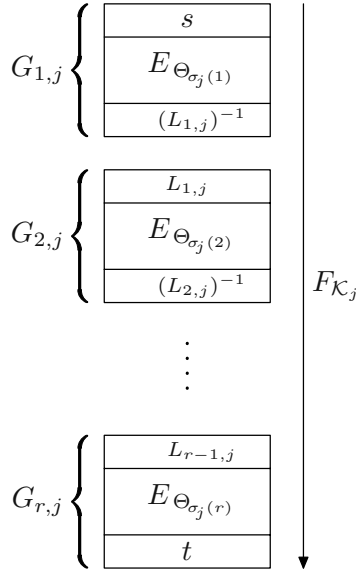


Fig. 3. Description $F_{K_j} = G_{r,j} \circ \dots \circ G_{2,j} \circ G_{1,j}$.

Now assign to each user j a private key K_j generated after the meta key K using a set of $r - 1$ linear one to one mappings $L_{1,j}, \dots, L_{r-1,j}$ from \mathbb{K}^n to itself, and a permutation σ_j of the set $\{1, \dots, n\}$. The user gets his key K_j as a list of functions $G_{1,j}, \dots, G_{r,j}$, which are provided as systems of n multivariate equations of homogeneous degree as described in Figs. 2 and 3.

A user initialization scheme needed to derive a user's key from the meta key K follows. From any input j one creates the permutation σ_j and the $r - 1$ one to one mappings $L_{i,j}$ by any pseudo- random generation mechanism or by any diversification algorithm.

We can now check that the users' functions F_{K_j} are distinct but equivalent descriptions of the meta function F_K . Indeed, for each user j , the one to one mappings at the end of $G_{k,j}$ and at the beginning of $G_{k+1,j}$ cancel out, and since the functions E_Θ are commuting, the effect of the permutation is annihilated.

3.3 Encryption and Decryption

In order to encrypt a digital content, the station may broadcast enabling block and cipher block pairs (EB_i, CB_i) produced with the help of any additional symmetric algorithm S_{CW} where the symmetric key is the control word generated as $CW := F_K(EB_i)$. Thus, the construction is given by $CB_i := S_{F_K(EB_i)}(B_i)$, where B_i denotes the content block. Now any user j can recover the content block by following a similar procedure, that is by computing $B_i := S_{F_{K_j}(EB_i)}^{-1}(CB_i)$.

3.4 Traitor Tracing Procedure

The procedure to identify traitors relies upon the two following claims which are substantiated in the security analysis given in the next section.

Claim 1. When the leakage originates from a single traitor l , the analysis of the description F' constructed by the traitor based on his description F_{K_l} allows the authority to decompose F' in r components G'_1 to G'_r such that $F' = G'_r \circ \dots \circ G'_1$. Moreover, each G'_i can be split as the composition of the functions G_i of the traitor and other “parasitic” functions which may differ from the identity function. Thus, the analysis reveals the order of composition of the functions G_i which in turn reveals the identity of the traitor through the knowledge of σ_l .

This first claim allows an authority provided with the meta key K to efficiently derive the permutation σ_l associated to the description F_{K_l} of the traitor from the leaked function F' , and thus to recover the identity l of the traitor.

Claim 2. When the leakage originates from a coalition of at most k traitors, the analysis of the description F' constructed by the k colluding traitors allows to decompose F' in r components G'_1 to G'_r such that the middle $r - 2\rho$ values come from “parasitized” functions G_i of a single traitor, for a well chosen ρ .

This second claim allows, by properly choosing the parameters of the system, specially ρ which exact definition is to be given in the next Section, to recover the identity of one of the traitors—say j —by deriving the values of the permutation σ_j on the set of integers $[\rho, r - \rho]$ from the values of the functions $G_{\rho,j}$ to $G_{r-\rho,j}$ alone. To achieve this goal, we must ensure that the middle part of the pirate description F' originates from the middle parts ρ to $r - \rho$ of one single traitor, while mixing traitors’ descriptions in the ranges $[1, \rho]$ and $[r - \rho, r]$ can still be tolerated.

4 Security Discussion

4.1 The IP Problem

The security of the proposed traceable iterated symmetric cipher relies to a large extent upon the security of special instances of the “Isomorphism of Polynomials” problem—hereafter called IP—namely the problem of *finding the hidden monomial* of the extended Matsumoto-Imai C^* scheme described in Section 3.1.

The IP problem with two secrets—see also [7,17]—consists in finding a pair (s, t) of one to one linear mappings between two sets A and B of multivariate polynomial equations of total degree d over a finite field \mathbb{K} . Denoting by $x = (x_1, \dots, x_n)$ an element of \mathbb{K}^n , we can write $y = A(x)$ as a system of polynomial equations:

$$\begin{cases} y_1 = P_1(x_1, \dots, x_n) \\ y_2 = P_2(x_1, \dots, x_n) \\ \vdots \\ y_n = P_n(x_1, \dots, x_n) \end{cases},$$

and similarly for B . In this setting the IP problem consists in finding a pair of one to one linear mappings s and t such that:

$$B(s(x)) = t(A(x)). \quad (2)$$

This problem is assumed to be difficult and it has been shown to be at least as hard as the “Graph Isomorphism” problem. Even for very special instances complexity remains high [2,6]. Note also that an efficient solution to the IP problem would lead to an efficient attack on SFLASHv2 [19] that has been selected by the European Nessie project.

4.2 Resisting Attacks against the Decryption Scheme

As explained in Section 2, the descriptions $F_{\mathcal{K}_j}$ of any user j must satisfy the usual security requirements of block ciphers. In particular, given any realistic number of input/output pairs of $F_{\mathcal{K}_j}$ corresponding to chosen or adaptively chosen input values, it must be computationally infeasible to infer any additional output value. Based on an investigation of the most natural attack strategies, we conjecture that this property is satisfied provided that:

1. Parameters q and n be chosen so that even if the monomial functions $E_{\Theta_1}, E_{\Theta_2}, \dots, E_{\Theta_n}$ can be guessed, solving the IP problem which consists of guessing s and t given a sufficient large number of input/output pairs of $F_{\mathcal{K}_j}$ be intractable. Based on the results in [7,2] we expect this condition to be satisfied provided that the complexity q^n of the best know attack be large enough, say at least 2^{80} . Since an enhanced attack of complexity $q^{n/2}$ is reported in the quadratic case in [7], an even more conservative choice would be to consider $q^n > 2^{160}$ in order to prevent a generalization of this attack to other instances of IP.
2. The value q^D , where D is the degree of the system of polynomial equations in n variables representing any $F_{\mathcal{K}_j}$ be large enough, say at least 2^{80} , to prevent attacks based on higher order derivation. Indeed, this would allow an attacker to predict one more output given an affine set of q^{D+1} input values and all but one of their corresponding outputs. D is about nq when r is large enough and q is the size of the finite field \mathbb{K} ;
3. The number of monomials of the system of n polynomial equations in n variables representing any $F_{\mathcal{K}_j}$, which is usually close to $n \binom{n+D-1}{D}$, be large enough to prevent an attacker from recovering the coefficients of this system using linear algebra and a sufficient number of input/output pairs of $F_{\mathcal{K}_j}$.

4.3 Tracing Single Traitor’s Pirate Description

We anticipate that in trying to produce an untraceable version of his description $F_{\mathcal{K}_j}$, a traitor j would adopt one of the following strategies:

1. Try to find one of the $r + 1$ one to one linear mappings $s, L_{1,j}, L_{2,j}, \dots, L_{r-1,j}$ and t , hidden to the attacker j . If an attacker j could recover one of these $r + 1$ linear mappings, say $L_{l,j}$, this would obviously allow him to incrementally recover all the $L_{i,j}$ for $i < l$, and all the $L_{i,j}$ for $i > l$, using the information provided by the mappings $G_{1,j}$ to $G_{r,j}$, and thus to recover the value of \mathcal{K}_j and to easily produce variants of his description $F_{\mathcal{K}_j}$ in an untraceable manner. Conversely, we conjecture this to be as hard as solving the IP problem of at least one of the $G_{i,j}$. The complexity of the best attacks reported in [7] are $O(q^n)$ in case $d > 2$ and $O(q^{n/2})$ in case $d = 2$.
2. Try to directly use the functions $G_{\cdot,j}$ without analyzing them, by modifying them so as to produce a concealed variant of the original description by composing the basic blocks $G_{\cdot,j}$ in the same order, but with “parasitic” functions whose effects eventually cancel out. That is the traitor tries to produce a sequence $(G'_{i,j})_{i \in [1,w]}$ with *two types* of blocks G' : those which can be written as $\varphi_i \circ G_{i,j} \circ \psi_{i+1}$ and those that do not rely on the available $G_{i,j}$ blocks and are denoted by Π_i . These data must be such that the effects of adding/composing the φ , Π and ψ mappings to the original blocks $G_{i,j}$ eventually cancel out, that is so that $F_{\mathcal{K}_j} = G'_{w,j} \circ \dots \circ G'_{1,j}$. (Please note that w can be greater than r because of the *second type* of blocks.) Also note that φ_i, ψ_i and Π_i have to be simple enough—for instance a reasonable number of monomials and a limited total degree—so that they could be easily constructed and efficiently computed.
3. Try to compose several blocks $G_{i,j}$ of his description. This attack is impossible as soon as the number of monomial in such composition is impractical. Since composition must be formally computed, $\binom{n+d-1}{d}$ terms must be formally put to the power of d which is quickly intractable. As will be seen in the sequel, composition of a small number of blocks $G_{i,j}$, say 2 of them, do not substantially complexify the tracing procedure. Therefore, only the composition of more than 3 blocks must be prevented.
4. Use a combination of any of the above strategies.

To trace traitor j from a pirate description G'_1, \dots, G'_w , the authority proceeds as follows. First, note that G'_1 is necessarily of the form $\psi_1 \circ L_{1,j} \circ E_{\Theta_{\sigma_j(1)}} \circ s$, that is of the *first type*. The authority thus searches for $\sigma_j(1)$ by using its knowledge of s^{-1} , and all the $E_{\Theta_i}^{-1}$: it computes $G'_1 \circ s^{-1} \circ E_{\Theta_i}^{-1}$ for each i , and guesses the right value i by testing the “simplicity” of the resulting function by means of chosen input/output pairs. The simplicity is evaluated by estimating the degree and the number of monomials. In case of a correct guess, the function has a low degree and a predetermined number of monomials whereas in case of a bad guess the function has terms of high degree. Having guessed the value $\sigma_j(1)$, we denote it by $\alpha(1)$.

The authority then has to get rid of terms of *second type* Π_i , until another term of *first type* is found. This is done again by evaluating the simplicity of the successive compositions:

$$\begin{aligned}
& G'_2 \circ G'_1 \circ s^{-1} \circ E_{\Theta_{\alpha(1)}}^{-1} \circ E_{\Theta_i}^{-1} , \\
& G'_3 \circ G'_2 \circ G'_1 \circ s^{-1} \circ E_{\Theta_{\alpha(1)}}^{-1} \circ E_{\Theta_i}^{-1} , \\
& \vdots
\end{aligned}$$

each time for all i until a simple composed function is found. The authority then finds the value $\sigma_j(2)$ and denotes it by $\alpha(2)$. The process goes on iteratively and eventually gives the permutation σ_j allowing the authority to trace traitor j .

While choosing the parameters of the system, we will make it hard for an attacker to formally compose two extended C^* blocks and totally intractable to compose three of them. The composition of two consecutive blocks can be easily thwarted since the above guessing procedure remains valid when replacing $E_{\Theta_i}^{-1}$ by $E_{\Theta_i}^{-1} \circ E_{\Theta_j}^{-1}$ varying both i and j at the same time, thus allowing to trace such compositions of two blocks as well.

4.4 Tracing k Traitors' Pirate Descriptions

The best collusion strategy we identified for a coalition of at most k traitors provided with distinct descriptions $F_{K_j} = G_{r,j} \circ \dots \circ G_{1,j}$ associated with the same meta description F_K is the following one.

The basic idea is that the traitors may take advantage of the fact that the initial mapping s and the final mapping t are identical for every user. This could allow them to detect a partial collision between their respective hidden permutation σ . Let us take the example of two traitors j and l searching for such a collision. They know their first blocks begin with the same mapping s , and if their first functions $E_{\sigma_j(1)}$ and $E_{\sigma_l(1)}$ were equal, then blocks $G_{1,j}$ and $G_{1,l}$ would be equal up to a one to one linear mapping, namely $L_{1,j}^{-1} \circ L_{1,l}$. Otherwise it would not be a one to one linear mapping. This is easy to test and provides a way for a pair of traitors to guess if their permutations take the same values on 1, *i.e.* if $\sigma_j(1) = \sigma_l(1)$.

Now, whether they succeed or not in the last step, the pair of traitors go further in the process by checking whether $G_{2,j} \circ G_{1,j}$ and $G_{2,l} \circ G_{1,l}$ are equal up to another hidden one to one linear mapping. (Remember that the commutativity of the functions E_{Θ_i} makes this possible.) In case of success, this would allow them to deduce that the images of the unordered set $\{1, 2\}$ under both permutations are equal: $\sigma_j(\{1, 2\}) = \sigma_l(\{1, 2\})$, and provide them with the value of $L_{2,j}^{-1} \circ L_{2,l}$. By iterating the process, the pair of traitors may identify any collision of their respective permutations on the set of integers $[1, t]$ for any $t \in [1, r]$, hereafter called a t -collision. Moreover, any detected t -collision provides a way to forge two new pirate descriptions by exchanging the first t components of their respective descriptions of the meta function F_K , as shown in Fig. 4.

Note that the traitors can search for all collisions. That is, when no u -collision was found for $u < t$, it still remains possible for them to find a t -collision, *when such a collision exists*. Of course, this scenario can be replayed with other traitor pairs, or even with the newly forged descriptions, leading to a possible great amount of untraceable pirate keys.

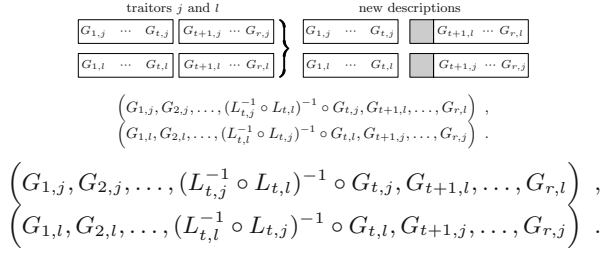


Fig. 4.

To avoid this situation, one encodes the identity of any user i in the values taken by the permutation σ_i on the middle interval $[\rho, r - \rho]$ of the original one $[1, r]$, for some well chosen $\rho < r/2$ so that the probability of *any* t -collision for $t \in [\rho, r - \rho]$ is arbitrarily small.

Obviously, attacks involving a single traitor can also be used by coalition of traitors in addition to the specific techniques discussed in this Section, but those can be handled the same way.

4.5 Non-detection and False Alarms

Let us derive the requirements the attack scenario of the previous Section puts on parameter ρ . First, for any traitors' pair, the probability that a t -collision holds is $1/\binom{r}{t}$. Thus the probability that a t -collision for a coalition of up to k traitors occurs for $t \in [\rho, r - \rho]$ is at most

$$P_k = \frac{k(k-1)}{2} \sum_{t=\rho}^{r-\rho} \frac{1}{\binom{r}{t}}.$$

At the same time, permutations of users must be distinguishable from their values in the interval $[\rho, r - \rho]$. This implies that the number of distinct identities available for the system will be at most $M = r!/(2\rho)!$.

Now if the scheme needs to handle at most N users where $N < M$, and assuming a coalition of up to k traitors, the probability of non-detection (the authority detects a collusion, but no matching identity is found) is given by $P_{k,\text{ND}} = (1 - N/M) P_k$ while the probability of false alarm (a wrong identity is pointed out) is given by $P_{k,\text{FA}} = N/M P_k$. This comes from the fact that there are $(M - N)$ permutations that do not correspond to any valid identity.

5 Practical Example

We provide realistic example parameters such that the scheme accommodates $N = 10^6$ users. The field of operation \mathbb{K} is taken to be $\text{GF}(2^{16})$ so that $m = 16$ and $q = 2^{16}$. Moreover, we chose $n = 5$ and the degree of the monomials in an extended C^* block to be $d = 4$. There is a total of 32 distinct $(d - 1)$ -tuples Θ such that $1 + q^{\theta_1} + \dots + q^{\theta_{d-1}}$ is prime to $q^n - 1$.

Letting $r = 32$ and $\rho = 13$ makes the probability of false alarms smaller than $2 \cdot 10^{-10}$ for any coalition of up to $k = 10$ traitors, smaller than $2.2 \cdot 10^{-8}$

for $k = 100$ traitors and smaller than $2.3 \cdot 10^{-6}$ for $k = 1000$. Probability of non-detection is smaller than $1.2 \cdot 10^{-7}$ when $k = 10$, smaller than $1.5 \cdot 10^{-3}$ when $k = 1000$. Other security requirements are met since $q^n = 2^{80}$ and furthermore the number of monomials in a building block of $F_{\mathcal{K}}$ is 350, so that in any formal composition of three of them the number of monomials is already more than $4 \cdot 10^6$, and in any formal composition of four blocks it is about 10^9 .

With this choice of parameters, the total size of any description equivalent to $F_{\mathcal{K}}$ is 21,8 KB. Speed of encryption is essentially determined by the number of multiplications in $F_{\mathcal{K}_j}$ to be performed and can roughly be estimated as follows: the 70 terms $x_1^{\nu_1} \cdots x_5^{\nu_5}$ of total degree four can be computed once for each block and then multiplied by the appropriate leading coefficients of the polynomials describing each output variable of a block. So one can compute the 70 homogeneous terms of degree 4 in 85 multiplications in \mathbb{K} and eventually compute y_1, \dots, y_5 in at most 5-70 multiplications in \mathbb{K} . Since there are 32 blocks, that makes a total of about 15000 multiplications to process any $F_{\mathcal{K}_j}$ on the 80 bit input. Additionally, the size of the overhead in this example is obviously 80 bits.

We propose another realistic set of parameters, hopefully more conservative, for applications where storage and speed of encryption are less critical concerns. The scheme handles up to $N = 10^6$ users. The field of operation is taken to be $\text{GF}(2^9)$, while the number of variables is set to $n = 19$ and the degree of the monomials is set to $d = 3$. There is a total of 190 distinct $(d - 1)$ -tuples Θ such that $1 + q^{\theta_1} + \dots + q^{\theta_{d-1}}$ is prime to $q^n - 1$. Choosing $r = 33$ and $\rho = 10$ makes the probability of false alarms smaller than $1.4 \cdot 10^{-19}$ for any coalition of up to $k = 10$ traitors, smaller than $1.52 \cdot 10^{-15}$ for any coalition of up to $k = 1000$ traitors, and the probability of non-detection smaller than $5 \cdot 10^{-7}$ for any coalition of up to $k = 10$ traitors, smaller than $5.4 \cdot 10^{-3}$ for any coalition of up to $k = 100$ traitors. Security requirements are met since $q^n = 2^{171}$ and the number of monomials in a building block of $F_{\mathcal{K}}$ is 25270, so that in any formal composition of three of them the number of monomials is already more than $90 \cdot 10^6$ and in any formal composition of three blocks it is already more than $3 \cdot 10^{13}$. In that case, the size of any equivalent decryption key is 916 KB. The 1330 monomials can be computed in 1520 multiplications in \mathbb{K} so that a building block requires 26790 multiplications and it takes about 900000 multiplications to evaluate any description $F_{\mathcal{K}}$ on the 171 bits of the input. The overhead is obviously of 171 bits.

6 Conclusion

A novel iterative block cipher which can be operated in a traceable manner has been introduced. The attacks investigated in our initial security analysis are easy to prevent by properly selecting system parameters. Improvements in these attacks are of course not precluded, since no reduction proof of the security to a well identified mathematical problem was found apart from obvious connection to the “Isomorphism of Polynomial” problem. Risks are obviously higher than for usual symmetric ciphers.

k	10	100	1000
$P_{k,FA}$	$< 2 \cdot 10^{-10}$	$< 2.2 \cdot 10^{-8}$	$< 2.3 \cdot 10^{-6}$
$P_{k,ND}$	$< 1.2 \cdot 10^{-7}$	$< 1.5 \cdot 10^{-5}$	$< 1.5 \cdot 10^{-3}$

$$N = 10^6, r = 32, \rho = 13, n = 5, \mathbb{F} = \text{GF}(2^8).$$

k	10	100	1000
$P_{k,FA}$	$< 1.4 \cdot 10^{-19}$	$< 1.5 \cdot 10^{-17}$	$< 1.6 \cdot 10^{-15}$
$P_{k,ND}$	$< 5 \cdot 10^{-7}$	$< 5.4 \cdot 10^{-5}$	$< 5.4 \cdot 10^{-3}$

$$N = 10^6, r = 33, \rho = 10, n = 19, \mathbb{F} = \text{GF}(2^9).$$

Fig. 5. Summary of parameters and corresponding probabilities.

Natural questions also arise: What security does the “Isomorphism of Polynomials” problem provide for small values of the number n of variables like those suggested in Section 5? Also, other building blocks could be considered, e.g. variants with two or more branches in each extended C^* block. Studying the effects of releasing the constraint that the monomial functions be distinct may lead to some performance improvements. We also note that since each user possesses an equivalent description, he is able to broadcast data to every other user. Besides traitor tracing, another interesting application of the proposed construction is whitebox cryptography [5]. Indeed, advantage can be taken from the fact that one can easily construct a huge number of equivalent descriptions, while those descriptions can be made arbitrarily large.

In its current shape, the proposed traceable block cipher has the advantage of being very insensitive to the maximum number of traitors tolerated while accommodating a large number of users. Due to its intrinsic block cipher structure and due to the fact that it does not generate any data expansion overhead, its implementation can be made very efficient.

References

1. Elad Barkan and Eli Biham, *In how many ways can you write Rijndael?*, available from the e-print at <http://eprint.iacr.org/2002/157/>.
2. Alex Biryukov, Christophe De Canniere, An Braeken, and Bart Preneel, *A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms*, Advances in Cryptology – EUROCRYPT 2003 (Eli Biham, ed.), Lecture Notes in Computer Science, vol. 2656, Springer-Verlag, 2003, pp. 33–50.
3. Dan Boneh and Matthew Franklin, *An Efficient Public Key Traitor Tracing Scheme*, Advances in Cryptology – CRYPTO ’99 (Michael Wiener, ed.), Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, 1994, pp. 338–353.
4. Benny Chor, Amos Fiat, and Moni Naor, *Tracing Traitors*, Advances in Cryptology – CRYPTO ’94 (Yvo G. Desmedt, ed.), Lecture Notes in Computer Science, vol. 839, Springer-Verlag, 1994, pp. 257–270.
5. Stanley Chow, Philip Eisen, Harold Johnson, and Paul C. Van Oorschot, *White-Box Cryptography and an AES Implementation*, Selected Areas in Cryptography – SAC 2002 (K. Nyberg and H. Heys, eds.), Lecture Notes in Computer Science, vol. 2595, Springer-Verlag, 2002, pp. 250–270.

6. Nicolas Courtois, Louis Goubin, Willi Meier, and Jean-Daniel Tacier, *Solving Underdefined Systems of Multivariate Quadratic Equations*, Public Key Cryptography – PKC 2002 (David Naccache and Pascal Paillier, eds.), Lecture Notes in Computer Science, vol. 2274, Springer-Verlag, 2002, pp. 211–227.
7. Nicolas Courtois, Louis Goubin, and Jacques Patarin, *C^{*-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai*, Advances in Cryptology – ASIACRYPT '98 (Kazuo Ohta and Dingyi Pei, eds.), Lecture Notes in Computer Science, vol. 1514, Springer-Verlag, 1998, pp. 35–49.
8. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir, *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, Advances in Cryptology – EUROCRYPT 2000 (Bart Preneel, ed.), Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, 2000, pp. 392–407.
9. Nicolas T. Courtois, Magnus Daum, and Patrick Felke, *On the Security of HFE, HFEv- and Quartz*, Public Key Cryptography – PKC 2003 (Yvo G. Desmedt, ed.), Lecture Notes in Computer Science, vol. 2567, Springer-Verlag, 2003, pp. 337–350.
10. Jean-Charles Faugère and Antoine Joux, *Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases*, Advances in Cryptology – CRYPTO 2003 (Dan Boneh, ed.), Lecture Notes in Computer Science, vol. 2729, Springer-Verlag, 2003, pp. 44–60.
11. Aviad Kipnis and Adi Shamir, *Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization*, Advances in Cryptology – CRYPTO '99 (Michael Wiener, ed.), Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, 1999, pp. 19–30.
12. Paul Kocher, Joshua Jaffe, and Benjamin Jun, *Differential Power Analysis*, Advances in Cryptology – CRYPTO '99 (Michael Wiener, ed.), Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, 1999, pp. 388–397.
13. Tsutomu Matsumoto and Hideki Imai, *Public Quadratic Polynomial-tuples for Efficient Signature Verification and Message Encryption*, Advances in Cryptology – EUROCRYPT '88 (Cristoph G. Günther, ed.), Lecture Notes in Computer Science, vol. 330, Springer-Verlag, 1988, pp. 419–453.
14. Moni Naor and Benny Pinkas, *Threshold Traitor Tracing*, Advances in Cryptology – CRYPTO '98 (Hugo Krawczyk, ed.), Lecture Notes in Computer Science, vol. 1462, Springer-Verlag, 1998, pp. 502–517.
15. Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt '88*, Advances in Cryptology – CRYPTO '95 (Vangalur S. Alagar and Maurice Nivat, eds.), Lecture Notes in Computer Science, vol. 963, Springer-Verlag, 1995, pp. 248–261.
16. Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, Advances in Cryptology – EUROCRYPT 1996 (U. Maurer, ed.), Lecture Notes in Computer Science, vol. 1070, Springer-Verlag, 1996, pp. 33–48.
17. Jacques Patarin, Louis Goubin, and Nicolas Courtois, *Improved Algorithms for Isomorphisms of Polynomials*, Advances in Cryptology – EUROCRYPT '98 (Kaisa Nyberg, ed.), vol. 1403, 1998, pp. 184–200.
18. Douglas R. Stinson and Ruizhong Wei, *Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes*, SIAM Journal on Discrete Mathematics **11** (1998), no. 1, 41–53.
19. *Specifications of SFLASH*, available from the site of the NESSIE workshop at <https://www.cosic.esat.kuleuven.ac.be/nessie/workshop/>.

A New Attack against Khazad

Frédéric Muller

DCSSI Crypto Lab, 18 rue du Docteur Zamenhof
F-92131 Issy-les-Moulineaux Cedex, France
`Frederic.Muller@m4x.org`

Abstract. Khazad is a new block cipher initially proposed as a candidate to the NESSIE project. Its design is very similar to Rijndael, although it is a 64-bit block cipher. In this paper, we propose a new attack that can be seen as an extension of the Square attack. It takes advantage of redundancies between the round key derivation and the round function, and also exploits some algebraic observations over a few rounds. As a result, we can break 5 rounds of Khazad faster than exhaustive key search. This is the best known cryptanalytic result against Khazad.

1 Introduction

Many recent block ciphers are built using an iterative Substitution Permutation Network (SPN). This includes in particular Shark [14], Square [5], Rijndael [6], Anubis [1] or Khazad [2]. These ciphers are generally designed to be immune against differential and linear cryptanalysis. However, a new powerful class of attack has emerged recently, the “Square” attack which was initially a dedicated attack [5] against the Square block cipher. It takes advantage of the bijectivity of most components of these ciphers (S-box, round key addition, ...), without analyzing their precise behavior. More generally, this class of high-level attacks can be seen as a dual technique to differential and linear cryptanalysis since it is based on the propagation of distributions along the cipher for a large set of plaintexts, rather than on statistical properties for a single plaintext (or a pair of plaintexts).

Since then, this technique has been successfully applied to many other block ciphers (see [3] and [8]). Currently, one of the best known attacks against Rijndael is Gilbert-Minier’s collision attack on 7-rounds [9] which can also be seen as an extension of the “Square” attack. Besides, a more generic name for this technique, namely the “integral” attack has been recently proposed [10]. We use this terminology in the present paper.

Khazad is a 64-bit SPN block cipher with 8 rounds. It offers several interesting features. First, it achieves full diffusion over one round using an MDS matrix layer. Furthermore, all components are involution, so the only difference between encryption and decryption lies in the key scheduling. Thus the same security is expected in both directions.

Khazad was initially proposed as a NESSIE [11] candidate for 64 bits block cipher. However, it was not selected due to his low security margin [12]. In the

Section 2, we provide some background about Khazad. Then, in Section 3 we present new observations about this cipher that we later exploit to mount a 5-round attack.

2 Some Background about Khazad

Khazad is a byte-oriented cipher. Indeed, all operations handle bytes of data: S-box, linear application over $GF(2^8)$, Like most word-oriented ciphers, Khazad may thus be subject to integral attacks. First, we describe quickly the main components of Khazad, then we present previously known cryptanalytic results against this cipher.

2.1 Description of the Cipher

We only give a short overview of Khazad. More details can be found in [2]. During encryption, it iterates 8 times a SP round function. Throughout this paper, we denote by P its linear layer and S its S-box layer. Thus, each round consists, in this particular order, of

- a S layer where a fixed S-box is applied to each byte of the current state.
- a P layer which consists of a square matrix in $GF(2^8)$ of size 8
- a XOR layer, using the corresponding round subkey. This layer is called X_i at round number i .

A first XOR layer is applied prior to the first round. Besides, the last round does not include a matrix layer. Thus the full encryption function can be written as

$$(X_8 \circ S) \circ (X_7 \circ P \circ S) \circ \cdots \circ (X_1 \circ P \circ S) \circ X_0$$

By convention, the notation “0.5 round” denotes either the first two layers of the round (the S and P layers together), or the XOR layer alone.

2.2 Previous Results about Khazad

All cryptographic results concerning Khazad are summarized in Table 1. The best known attack so far was the straightforward application of the integral attack, originally proposed by the designers of the cipher [2]. Indeed, if a set of 256 plaintexts is introduced, such that the first byte takes all 256 possible values while other bytes have constant values, distributions for each byte can be easily described over 2 rounds (see Figure 1). For each byte, the following notations are used to represent these distributions over this set of 256 plaintexts

- A represents bytes where “All” possible values are represented exactly once.
- C represents bytes which have a “Constant” value
- S represents bytes where the “Sum” of all values is 0.

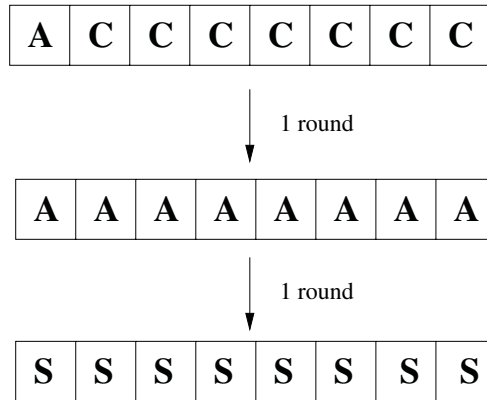


Fig. 1. The straightforward integral attack.

Table 1. Summary of Known Attacks Against Khazad.

<i>Type of Attack</i>	<i>Rounds</i>	<i>Time</i>	<i>Data</i>
integral attack [2]	3	2^{16}	2^9
impossible differential [12]	3	2^{64}	2^{13}
integral attack [2]	4	2^{80}	2^9
weak keys [4]	5	2^{43}	2^{38}
improved integral (this paper)	5	2^{91}	$\simeq 2^{64}$

From Figure 1, it appears that all bytes have a S distribution after 2 rounds. Such balanced distributions provide a 2-round distinguisher. Since there is no matrix layer in the last round, the corresponding subkey bytes can be guessed separately to mount a 3-rounds attack against Khazad. The resulting complexity is roughly 2^{16} S-box lookups and 2^9 chosen plaintexts. Besides, this attack can be directly extended to 4 rounds by guessing one additional subkey. This increase the time complexity by a factor 2^{64} .

Other attacks have been examined throughout the NESSIE evaluation process. An impossible differential attack exists on 3 rounds of Khazad but its complexity is larger than the integral attack [12]. The design rationale apparently prevents differential and linear attacks since a large number of S-boxes is activated at each round. Besides, Gilbert-Minier's attack on Rijndael does not apply very well here, since it requires partial collision. New ideas to attack involutinal ciphers have been recently proposed [4]. Indeed the cycle structure of 5-rounds Khazad presents some surprising properties. However these observations do not result yet on a concrete attack. Finally, the only cryptanalytic result on 5-rounds Khazad is the class of 2^{64} weak keys identified in [4] which can be broken with 2^{43} steps of analysis using 2^{38} encryption blocks.

3 New Observations on Khazad

In this Section, we investigate some properties of Khazad. Our first observations concern the key scheduling, which is a Feistel network based on the round function. We show some surprising weaknesses resulting from this redundancy. Then, we describe some algebraic properties of the cipher over a reduced number of rounds.

3.1 Redundancies in the Round Key Derivation

In order to speed up Khazad while keeping the same security, its designers adopted a key scheduling that inherits many properties of the round function. While this key scheduling can be viewed as a simple Feistel network, the derivation of the i -th round key K_i is basically just one round of encryption applied to K_{i-1} , with a particular XOR layer. Initially, the actual 128 bits of secret key are splitted into K_{-1} and K_{-2} which serve as initial values. Thus, we have the following relation, for $0 \leq i \leq 8$,

$$K_i = P \circ S(K_{i-1}) \oplus C_i \oplus K_{i-2} \quad (1)$$

where the C_i 's are round constants. The use of the round function during key scheduling creates surprising cascade eliminations during encryption. To illustrate this, we consider encryption of the plaintext

$$Plain = K_0 \oplus S \circ P(0)$$

which depends only on the first subkey K_0 . After 1 round, the internal value (denoted as Iv_1) is

$$\begin{aligned} Iv_1 &= K_1 \oplus P \circ S(Plain \oplus K_0) \\ &= K_1 \oplus P \circ S \circ S \circ P(0) \\ &= K_1 \end{aligned}$$

since P and S are involution. Then, after the second round, the internal value Iv_2 is

$$Iv_2 = K_2 \oplus P \circ S(Iv_1) = K_2 \oplus P \circ S(K_1) = K_0 \oplus C_2$$

because of relation (1). Thus, Iv_2 is basically known when K_0 is known, independently of K_1 and K_2 . The following S and P layers can also be included, so we obtain an internal value depending only on K_0 after 2.5 rounds.

Many similar eliminations can be obtained with chosen plaintext or ciphertext once a round key is known (or guessed). Obviously, this observation suggests guessing K_0 to obtain a known intermediate value and then trying to extend these observations over the last rounds. We describe such an attack in Section 4.

3.2 Algebraic Properties of 2.5 Rounds of Khazad

In this section, we consider the algebraic properties of Khazad. More precisely, we show that the last 2.5 rounds of Khazad can be expressed using a reduced

number of algebraic relations. We also show that this system can be used to retrieve several subkey bits, once a few intermediate values and their corresponding ciphertexts are known.

As it was originally argued in [10], interpolation and algebraic attacks are good candidates to be combined with an integral attack which usually provides known intermediate values (or linear relations between these values). In the case of Khazad, we consider the following situation, encountered later in Section 4.

- We know 256 full intermediate values: Y_1, \dots, Y_{256} .
- 2.5 rounds of encryption remain unknown. The 3 corresponding round keys are denoted as K , K' and K'' .
- The resulting ciphertexts Z_1, \dots, Z_{256} are known. The last 2.5 rounds of encryption can be expressed as

$$Z_i = K'' \oplus S(K' \oplus P \circ S(K \oplus Y_i))$$

or, equivalently,

$$S(Z_i \oplus K'') = K' \oplus P \circ S(K \oplus Y_i)$$

for $1 \leq i \leq 256$. Then, the first byte of $S(K \oplus Y_i)$, later referred to as w_i can be obtained by just guessing the first byte of K . Let $P_1(x)$ denotes the linear function that returns the first byte of $P(x)$ for any 64 bits input x . We have the following relation

$$P_1 \circ S(Z_i \oplus K'') = P_1(K') \oplus w_i \quad (2)$$

In addition, if we guess the byte $P_1(K')$, we obtain, for each i , a condition on K'' of the form

$$P_1 \circ S(\text{known} \oplus K'') = \text{known} \quad (3)$$

While it is not straightforward to solve such a non linear system, we apparently obtain enough conditions to retrieve the value of K'' .

Suppose we replace, in relation (3), the S-box by its exact algebraic interpolation over $\text{GF}(2)$. From the left hand side of (3), one sees that only a reduced number of monomials in the bits of K'' appear. Indeed, S operates on the bytes of K'' , thus the monomials are those involving bits of K'' that “belong” to the same byte. For instance, representing K'' as (k_1, \dots, k_{64}) , all monomials over (k_1, \dots, k_8) may appear while no monomial involving simultaneously k_8 and k_9 can appear.

Thus, (3) can be seen as a system of 8 relations over $\text{GF}(2)$ involving $8 \times 2^8 = 2^{11}$ monomials in the bits of K'' . Since 256 such relations are known (one for each $i = 1, \dots, 256$), we obtain a system with 2^{11} unknown monomials and 2^{11} relations. Two bytes of round keys have been guessed to build this system.

3.3 Properties of the Linear System

In the previous Section, we built a linear system over $\text{GF}(2)$ of 2^{11} relations involving 2^{11} unknowns, which are monomials over the 64 bits of the last round subkey. This can be summarized as

$$b = M x$$

where x and b are vectors of 2^{11} bits and M is a square matrix obtained after replacing the S-box by its algebraic expression over $\text{GF}(2)$. x contains the unknown monomials and, for $i = 0, \dots, 256$, each relation (3) is turned into eight conditions on the bits of x , that correspond to bits b_{8i}, \dots, b_{8i+7} of b . More precisely, from relation (2), we see that, for all i ,

$$b_i = c_i \oplus \{P_1(K')\}_i$$

where $\{P_1(K')\}_i$ denotes the bit number $(i \bmod 8)$ of $P_1(K')$, and c_i depends only on the intermediate values and the first byte of K .

In the general case, one could expect to solve this system by inverting the matrix M . However, M is built from an S-box interpolation, thus it is not a random matrix. It turns out that rows of M cannot have full rank for two reasons:

- The algebraic degree of the Khazad S-box is 7, therefore the 8 columns of M corresponding to monomials of degree 8 necessarily contain only zeroes.
- The coefficients of M corresponding to degree 7 monomials are independent of the plaintext.

Indeed, every output bit s_j of the S-box can be represented by a relation of the form

$$s_j = \sum_{\alpha_1 + \dots + \alpha_8 \leq 7} \beta_j^{(\alpha_1, \dots, \alpha_8)} i_1^{\alpha_1} \dots i_8^{\alpha_8} \quad (4)$$

for some coefficients $\beta_j^{(\alpha_1, \dots, \alpha_8)}$, with (i_1, \dots, i_8) denoting the input bits.

However, M is obtained by applying several times the S-box to inputs of the form

$$(i_1, \dots, i_8) = (c_1 \oplus k_1, \dots, c_8 \oplus k_8)$$

where the c_i 's are ciphertext bits and the k_i 's are subkey bits. Substituting these values in (4), it is clear that terms of degree 7 in the subkey bits are independent of the c_i 's. Therefore, for all $i = 0, \dots, 256$, relation (3) always provides the same coefficients for degree 7 monomials. The 64 corresponding columns of M are not free (and in fact have rank 8).

Moreover, when computing $b_t \oplus b_{8i+t}$ for $i = 1, \dots, 255$ and $t = 0, \dots, 7$, monomials of degree 7 are eliminated. Hence, we can obtain $8 \times 255 = 2040$ new relations of degree 6, thus involving only $2^{11} - 8 - 64 = 1976$ monomials. This result on a new matrix M' having 2040 lines and 1976 columns. The initial system

$$b = M x$$

can be rewritten as

$$b' = M' x'$$

where the vector b' contains 2040 bits of the form $b_t \oplus b_{8i+t}$ and x' contains the 1976 monomials of degree 6 or less. Besides, b' does not depends on K' , whose

bits get eliminated when computing $b_t \oplus b_{8i+t}$. A direct application of the gauss algorithm on M' provides at least 64 conditions on its rows, thus conditions on the bits of b' . These conditions must be satisfied when the correct byte of K has been guessed.

Therefore, we do not have to solve the initial system. From the interpolation matrix M , we can build $2040 - 1976 = 64$ linear conditions and thus detect the correct guess for the corresponding 8 bits of K . We programmed this algebraic step using the NTL library [13]. It turns out from our experiment that the kernel of M' has always rank 64 (although it would be no problem if its dimension was larger). Thus we obtain easily enough linear conditions to verify the correct guess.

To summarize, we have shown that the last 2.5 rounds of Khazad can be expressed with a low degree algebraic system, after guessing a reduced number of bits. 64 linear conditions can be used to discard wrong guesses without actually solving this system.

4 An Attack against 5 Rounds of Khazad

In this Section, we develop the previous observations on Khazad to mount a new attack against 5 rounds of this cipher. The sketch of this attack works as follows

4.1 Sketch of the Attack

- Guess all 64 bits of K_0
- Guess 8 bits of K_1
- Introduce 256 chosen plaintexts in order to
 - apply the integral attack, starting from the end of the X_0 layer
 - obtain known intermediate values after 2.5 rounds as in Section 3.1
- Build the interpolation matrix as described in Section 3.2.
- Build 64 linear conditions from the matrix.
- Guess the first byte of K_3
 - Verify the linear conditions.
 - Discard wrong guesses.
- A large portion of guess of K_0 and K_1 are also discarded through the absence of a matching K_3
- Recover the whole secret key.

Most elements in this attack have been developed previously. Additional elements needed to connect all together are described in the following section.

4.2 Strengthening the Integral Attack

Once K_0 has been guessed, we can choose the plaintext $Plain$ to obtain any intermediate value after 0.5 round, since this value is equal to $P \circ S(Plain \oplus K_0)$ (and also to $K_1 \oplus Iv_1$). We consider an integral attack starting from there. Let us consider the set of 256 plaintexts such that $Iv_1 \oplus K_1$ takes all values on its

first byte and has constant value equal to 0 on its other bytes. This can be represented as

$$Iv_1 = K_1 \oplus (i, 0, 0, 0, 0, 0, 0, 0)$$

for $0 \leq i \leq 255$. As in the classical integral attack, we obtain A distributions after 1.5 round and S distributions after 2.5 rounds. Besides, since the first byte of K_1 is guessed, we have, for all i

$$Iv_2 = K_2 \oplus P \circ S(K_1 \oplus (i, 0, \dots, 0)) = K_0 \oplus C_2 \oplus P(\Delta_i)$$

where Δ_i is known. Hence, we obtain 256 known intermediate values after 2.5 rounds. Moreover these values are balanced as in the integral attack of [2] though we do not specifically use this property.

Then, we are exactly in the situation described in Section 3.2 with 256 known intermediate values and the corresponding ciphertexts, with 2.5 rounds in-between. We have seen that a matrix can be built and 64 linear conditions derived from this matrix. Using them, we can guess then verify the value of the first byte of K_3 . Since there are 64 conditions, many wrong guess on K_0 and K_1 can even be filtered out by the absence of a matching value for K_3 . The number of remaining guesses afterwards is only

$$2^{64} \times 2^8 \times 2^8 \times 2^{-64} = 2^{16}$$

thus we can guess the 56 remaining bits of K_1 and deduce the full secret key - which is equivalent to (K_0, K_1) - for a total complexity of 2^{80} basic operations.

In fact, the linear algebra step has a larger complexity. The matrix we build is independent of the 8 guessed bits of K_1 , so we need to build it 2^{64} times, and then apply the Gaussian algorithm in each case. This algorithm has complexity of $(2^{11})^3$ binary operations. Using 32 bits instructions, it can be fasten up to obtain a complexity equivalent to 2^{28} S-box lookups. Thus, this step is roughly equivalent to $2^{64} \times 2^{28} = 2^{92}$ S-box lookups.

On the other hand, building the matrix of interpolation has a much smaller complexity since it can be largely precomputed (it is just a collection of smaller matrix blocks, each depending on 8 bits of ciphertext). Besides, the cost of verifying linear conditions corresponds in average to $2^{72} \times 2^8 \times 2 = 2^{81}$ evaluations of linear conditions on 2^{11} bits long vectors, each costing roughly 2×2^{11} bitwise operations. Using 32 bits instruction, this is roughly equivalent to

$$2^{81} \times 2^{11} \times 2 \times 2^{-5} = 2^{88}$$

S-box lookups. Therefore, the dominant cost in our attack is the linear algebra step.

To summarize, our attack against 5 rounds Khazad recovers the full 128 bits secret key with time complexity equivalent on average to 2^{91} S-box lookups and using basically the complete dictionary of 2^{64} plaintexts.

4.3 Overview of Cryptanalytic Results against Khazad

In Table 1, we have summarized all known cryptanalytic results against reduced-round versions of Khazad. Our improved integral attack is the best cryptanalytic result against Khazad. However, its data complexity represents in average the complete dictionary of 2^{64} plaintexts. Indeed, we need to encrypt 256 plaintexts for each guess of the first subkey. It is possible that the correct subkey is identified early, however, in average, all possible plaintexts will have been encrypted by the time we find the correct K_0 . We did not manage to find a technique to guess the subkeys in a better order, or to trade data complexity for time complexity. This is a topic for further research.

In practice this huge data complexity will make the attack infeasible, although it is significantly faster than exhaustive key search. Furthermore, it is widely considered that recent block ciphers should resist key recovery attack even when the full dictionary is known. Therefore, we consider this new attack is a significant step forward in the analysis of Khazad. Whether it can be extended to 6 or more rounds remains an open question that should be further investigated.

5 Possible Extensions

The attack we have described in Section 4 does not depend in depth from the components of this cipher. Concerning the S-box, the only property we use is its algebraic degree of 7. Concerning the MDS matrix, no property is specifically used. Therefore, Khazad cannot be strengthened by changing these components, and our attack depends only on the high-level structure of the cipher.

5.1 Key Scheduling Redundancy Attacks

In the case of Khazad, we have shown that re-using the round function inside the key scheduling has surprising effects. More generally, when a block cipher E uses in its key scheduling the same basic components as in the round function, a general problem is to consider the encryption of a chosen plaintext

$$Plain = \Phi(K_0)$$

for a well chosen function Φ of the first round key K_0 . More precisely, one should investigate if a cascade elimination cannot occur and yield a predictable value of $E_K(Plain)$, or even a simple function of a subkey. For instance, if

$$E_K(Plain) = \Psi(K_i)$$

for some i and some function Ψ , one may recover K_i from the guess of K_0 with time and data complexity roughly equivalent to the size of the subkeys. If, in addition, the full secret key can be reconstructed from K_0 and K_i (which is sometimes the case for key scheduling based on Feistel networks), this can lead

to an attack. This threat mostly concerns “small” block ciphers (like Khazad), where the round subkeys are smaller than the secret key.

In addition, an improvement would be to guess only a part of the first subkey, to obtain partially known intermediate values, in the case of SPN block ciphers that do not achieve full diffusion. This is the case of Anubis or Rijndael, though we did not manage to obtain any such observation against those. Furthermore, the existence of improved cascade eliminations on Khazad should also be further analyzed. More generally, using a key scheduling that is not too similar to the round function is probably a more reasonable thing.

5.2 Combining Integral and Interpolation Attacks

This idea of combining integral attacks with attacks based on the algebraic properties of a block cipher was originally introduced in [10]. However, no successful application has been reported since then. Our improved integral attack against Khazad is apparently the first successful combination of these two cryptanalytic techniques, although we also use additional properties of Khazad here.

The problem is that integral attacks generally end up providing some information concerning a balanced set of intermediate values. This type of property does not pass well across S-box layers, while the diffusion layers very quickly increase the number of monomials. Thus it is generally difficult to write simple algebraic relations, even after guessing some subkey bits as we did for Khazad. An other specific problem is that only algebraic relations where intermediate values are expressed as a function of subkey and ciphertext bits are generally useful for interpolation attacks. For instance, low degree algebraic relations from the inversion in $GF(2^8)$ cannot be used, at least in a straightforward manner. In spite of these problems, we believe such attacks combining different cryptanalytic techniques may be of interest in the future.

5.3 Other Algebraic Approaches

In Section 3.3, we obtained a large multivariate, non linear system over $GF(2)$. We used the relinearization technique [15], that means replacing all monomials (which happen to be present in reduced number here) by new unknowns and apply usual linear algebra techniques. This technique is not the best method known to solve nonlinear multivariate systems. However, it turns out to be sufficient and quite successful since we can obtain simple conditions on subkey bits by reducing the underlying matrix. In fact, we do not even need to solve this system, to finish with.

In a very generic way, what we obtain, for each guess of K_0 , is a system of low degree involving a few unknown subkey bits. We need either to solve this system, or to detect quickly if it has some solutions. Our attack uses the second strategy, and requires one application of the gauss algorithm on a 2^{11} bits square matrix. In the light of recent progress ([7], [16]), better techniques to directly solve the system could be considered. However, it seems unlikely the time complexity could be pushed below the length of the outside loop, namely

2^{64} . Thus any complexity gain would probably be limited. However, it would be interesting to find if a similar simple system over more than 2.5 rounds could be derived.

5.4 Exposure of Round Keys

It results from the previous observations that the security of 5 rounds of Khazad depends only on the secrecy of the first subkey. Indeed the complexity of our attack is quite high, especially the data complexity, however this is mostly due to the cost of guessing the first subkey K_0 .

If, somehow, the first round subkey is exposed, then 5 rounds reduced Khazad becomes insecure. In this case, the complete secret key can be recovered by applying a few times the attack of Section 4, which has complexity of only 2^{28} S-box lookups and 256 chosen plaintexts when K_0 is known. We consider this property is quite undesirable. Indeed, information about the first round subkey may be obtained by other means than exhaustive search. For instance, side channel attack techniques may provide this kind of information.

6 Conclusion

We have proposed a new attack against the block cipher Khazad. This cipher is very interesting, because it constitutes a reduced and simplified version of Rijndael, so its analysis is very helpful in understanding the security of word-oriented SPN block ciphers, which are now largely used since the standardization of Rijndael as the AES. In particular, the new class of integral (aka Square) attacks which are (almost) independent of the S-boxes should be further investigated.

In this paper, we break 5 rounds of Khazad (against 8 rounds for the full cipher) faster than exhaustive search: we use about 2^{64} chosen plaintexts and 2^{91} S-box lookups in average. Although the cryptanalytic techniques we exploit are not new, we combine them in a new and unexpected way to improve on known attacks. A very surprising improvement arises from some redundancies between the key scheduling and the round function of Khazad. Whether this attack can be improved or extended to 6 rounds remains a topic for further research.

References

1. P. Barreto and V. Rijmen. The Anubis Block Cipher. In *First Open NESSIE Workshop, KU-Leuven*, 2000. Submission to NESSIE.
2. P. Barreto and V. Rijmen. The Khazad Legacy-Level Block Cipher. In *First Open NESSIE Workshop, KU-Leuven*, 2000. Submission to NESSIE.
3. P. Barreto, V. Rijmen, J. Nakahara Jr., B. Preneel, J. Vadevalle, and H. Y. Kim. Improved SQUARE Attacks against Reduced-Round HIEROCRYPT. In M. Matsui, editor, *Fast Software Encryption – 2001*, volume 2355 of *Lectures Notes in Computer Science*, pages 165–173. Springer, 2001.

4. A. Biryukov. Analysis of Involutorial Ciphers: Khazad and Anubis. In T. Johansson, editor, *Fast Software Encryption – 2003*, Lectures Notes in Computer Science. Springer, 2003. To appear.
5. J. Daemen, L. Knudsen, and V. Rijmen. The Block Cipher Square. In E. Biham, editor, *Fast Software Encryption – 1997*, volume 1267 of *Lectures Notes in Computer Science*, pages 149–165. Springer, 1997.
6. J. Daemen and V. Rijmen. AES Proposal: Rijndael. In *AES Round 1 Technical Evaluation CD-1: Documentation*. NIST, 1998.
7. J.C. Faugère and A. Joux. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In D. Boneh, editor, *Advances in Cryptology – Crypto’03*, volume 2729 of *Lectures Notes in Computer Science*, pages 44–60. Springer, 2003.
8. N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting. Improved Cryptanalysis of Rijndael. In B. Schneier, editor, *Fast Software Encryption – 2000*, volume 1978 of *Lectures Notes in Computer Science*, pages 213–230. Springer, 2000.
9. H. Gilbert and M. Minier. A Collision Attack on Seven Rounds of Rijndael. In *Third AES Conference*, pages 230–241. NIST, 2000.
10. L. Knudsen and D. Wagner. Integral Cryptanalysis. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption – 2002*, volume 2365 of *Lectures Notes in Computer Science*, pages 112–127. Springer, 2002. Extended Abstract.
11. NESSIE - New European Schemes for Signature, Integrity and Encryption. <http://www.cryptonessie.org>.
12. NESSIE Security Report D20, version 2-0. Available at <http://www.cryptonessie.org>.
13. NTL library. Available at <http://www.shoup.net>.
14. V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. De Win. The Cipher SHARK. In D. Gollmann, editor, *Fast Software Encryption – 1996*, volume 1039 of *Lectures Notes in Computer Science*, pages 99–112. Springer, 1996.
15. A. Shamir and A. Kipnis. Cryptanalysis of the HFE Public Key Cryptosystem. In M. Wiener, editor, *Advances in Cryptology – Crypto’99*, volume 1666 of *Lectures Notes in Computer Science*, pages 19–30. Springer, 1999.
16. A. Shamir, J. Patarin, N. Courtois, and A. Klimov. Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations. In B. Preneel, editor, *Advances in Cryptology – Eurocrypt’00*, volume 1807 of *Lectures Notes in Computer Science*, pages 392–407. Springer, 2000.

An Efficient Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack^{*}

Chong Hee Kim, Yong Ho Hwang, and Pil Joong Lee

IS Lab, Dept. of Electronic and Electrical Eng., POSTECH, Korea
{chhkim,yhhwang}@oberon.postech.ac.kr, pjl@postech.ac.kr
<http://islab.postech.ac.kr>

Abstract. We propose a new public key trace and revoke scheme secure against adaptive chosen ciphertext attack. Our scheme is more efficient than the DF scheme suggested by Y. Dodis and N. Fazio[9]. Our scheme reduces the length of enabling block of the DF scheme by (about) half. Additionally, the computational overhead of the user is lower than that of the DF scheme; instead, the computational overhead of the server is increased. The total computational overhead of the user and the server is the same as that of the DF scheme, and therefore, our scheme is more practical, since the computing power of the user is weaker than that of the server in many applications. In addition, our scheme is secure against adaptive chosen ciphertext attack under only the decision Diffie-Hellman (DDH) assumption and the collision-resistant hash function H assumption, whereas the DF scheme also needs the *one-time* MAC (message authentication code) assumption.

1 Introduction

A broadcast encryption scheme enables a center to send encrypted data to a large group of users over an insecure channel, where only legitimate users can decrypt the data. The set of legitimate users is dynamically changing, so it should be possible to prevent some revoked users from decrypting the data. The broadcast encryption scheme has numerous applications, such as pay-TV systems, the distribution of copyrighted materials, internet multicasting of video, music, magazines, and so on.

A. Fiat and M. Naor first formalized the basic definitions and paradigms of the broadcast encryption scheme [11]. Afterwards, many variants have been investigated. One example is the scheme of tracing traitors [6]. In this setting, the center can trace the traitors after a pirate decoder is confiscated. There are two types of approaches to the traitor-tracing scheme. One is a scheme that uses a secret key and coding approach [4,6,12,16,17,18,19] and the other uses a public key [3,14]. In the secret key scheme, the keys in the pirate decoder can

^{*} This research was supported by University IT Research Center Project, the Brain Korea 21 Project, and Com2MaC-KOSEF.

be identified by combinatorial methods. In the public key approach, the size of the enabling block is independent of the number of subscribers. In addition, the public key traitor tracing schemes enable the center to prepare a public key that allows any entity to broadcast data to users. There is another variant of broadcast encryption, the revoke system, which concentrates on the problem of excluding a certain subset of users from receiving the data in a dynamically changing set of users. There are many revoke systems that use the secret key setting. These schemes are also divided into two categories. One is for stateless receivers [15,13,2] and the other is for non-stateless receivers [21,22].

Recently, a public key traitor-tracing scheme with the revocation capability was introduced by W. Tzeng and Z. Tzeng [20]. They also proposed a variant of their basic scheme to be secure against *adaptive chosen ciphertext attack* (CCA2). However, Dodis and Fazio noted that W. Tzeng and Z. Tzeng's scheme was *not* secure against CCA2 even if a single user is corrupted [9]. Dodis and Fazio also proposed their own scheme secure against CCA2 under the decision Diffie-Hellman (DDH) assumption, the collision-resistant hash function H assumption, and the *one-time* MAC assumption [9].

Our Results. We propose a new public key trace and revoke scheme secure against CCA2. Our scheme does not use the additional *one-time* MAC, so its security does not depend on the *one-time* MAC assumption. The length of the enabling block of our scheme is about half that of the DF scheme. Additionally, the computational overhead of the user is lower than that of the DF scheme instead the computational overhead of the server is increased. The total computational overhead of the user and the server is the same as that of the DF scheme. (We only consider the computation of exponentiation computed by the server and the user. If we did the analysis more precisely, our scheme is more efficient than the DF scheme because it does not require computational overhead for the MAC). Our scheme is more practical, since the computing power of the user is weaker than that of the server in many applications.

By slightly modifying standard tracing algorithms from previous schemes (e.g. [20]), our scheme can be a fully functional trace and revoke scheme. However, due to space limitations we will omit the tracing part and focus only on the revoke scheme, which is the original contribution of this paper.

2 Preliminaries

In this section, we review the Lagrange interpolation in the exponent, the decision Diffie-Hellman (DDH) assumption, and public key encryption schemes secure against CCA2.

The Lagrange Interpolation in the Exponent. Let q be a prime and $f(x) = \sum_{t=0}^z a_t x_t$ a polynomial of degree z over Z_q . Let x_0, \dots, x_z be distinct elements in Z_q . Then using the Lagrange interpolation, we can express $f(x)$

as $\sum_{t=0}^z (f(x_t) \cdot \lambda_t(x))$, where $\lambda_t(x) = \prod_{0 \leq j \neq t \leq z} \frac{x_j - x}{x_j - x_t}$, $0 \leq t \leq z$. We define the Lagrange interpolation operator as: $LI(x_0, \dots, x_z; f(x_0), \dots, f(x_z))(x) = \sum_{t=0}^z (f(x_t) \cdot \lambda_t(x))$.

Next, we consider a cyclic group G of order q and a generator g of G . Let $v_t = g^{f(x_t)}$, $0 \leq t \leq z$, where $x_t \in Z_q$ and $v_t \in G$. Then we define the Lagrange interpolation operator in the exponent as: $EXP-LI(x_0, \dots, x_z; v_0, \dots, v_z)(x) = g^{LI(x_0, \dots, x_z; f(x_0), \dots, f(x_z))} = \prod_{t=0}^z g^{(f(x_t) \cdot \lambda_t(x))} = \prod_{t=0}^z v_t^{\lambda_t(x)}$. We also remark that $EXP-LI(x_0, \dots, x_z; v_0^r, \dots, v_z^r)(x) = [EXP-LI(x_0, \dots, x_z; v_0, \dots, v_z)(x)]^r$. In what follows, we will refer to a function of the form $g^{f(x)}$, where $f(x)$ is polynomial, as an EXP -polynomial.

The Decision Diffie-Hellman Assumption. Let G be a group of large prime order q , and consider the following two distributions:

- the distribution \mathbf{R} of random quadruples $(g_1, g_2, u_1, u_2) \in G^4$,
- the distribution \mathbf{D} of quadruples $(g_1, g_2, u_1, u_2) \in G^4$, where g_1, g_2 are random, and $u_1 = g_1^r$ and $u_2 = g_2^r$ for random $r \in Z_q$.

The decision Diffie-Hellman (DDH) assumption is that it is computationally hard to distinguish these two distributions. That is, we consider an algorithm that should output 0 or 1, given a quadruple coming from one of the two distributions. Then the difference between the probability that it outputs a 1 given an input from \mathbf{R} , and the probability that it outputs a 1 given an input from \mathbf{D} is negligible.

Our scheme is based on the modified Cramer-Shoup (M-CS) scheme [5] and the DF scheme is based on the Cramer-Shoup (CS) scheme [7]. The M-CS scheme is a variant of the CS scheme. We briefly review these schemes.

The Cramer-Shoup Scheme. Given a security parameter 1^λ , the secret key is (x_1, x_2, y_1, y_2, z) and the public key is $(p, q, g_1, g_2, c, d, h, H)$, where p is a λ -bit prime, g_1, g_2 are generators of G (a subgroup of Z_p^* of a large prime order q), function H is a hash function chosen from a collision-resistant hash function family, $x_1, x_2, y_1, y_2, z \xleftarrow{R} Z_q$, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$.

Given a message $m \in G$, the encryption algorithm runs as follows. First, it chooses $r \xleftarrow{R} Z_q$ and computes $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r m$, $\alpha = H(u_1, u_2, e)$, $v = c^r d^{\alpha}$. The ciphertext is (u_1, u_2, e, v) . Given a ciphertext, the decryption algorithm runs as follows. First, it computes $v' = u_1^{x_1 + y_1 \alpha} \cdot u_2^{x_2 + y_2 \alpha}$. Next, it performs a validity check. If $v \neq v'$, then it outputs an error message, denoted ' \perp '; otherwise, it outputs $m = \frac{e}{v'}$. The security of this scheme against CCA2 is proven, based on DDH assumption, in [7].

The Modified Cramer-Shoup Scheme. R. Canetti and S. Goldwasser slightly modified the above CS scheme as follows, without losing in security [5]. If the decryption algorithm finds $v \neq v'$, instead of outputting ' \perp ' it outputs

a random value in G . In a sense, the modified scheme is even “more secure” since the adversary is not notified by the decryption algorithm whether a ciphertext is valid.

Now that the decryption algorithm does not explicitly check validity, given (u_1, u_2, e, v) it outputs $(\frac{e}{u_1}) \cdot (\frac{v'}{v})^s$ instead, where v' is computed as in the CS scheme and $s \xleftarrow{R} Z_q$. Note that the decryption algorithm is now randomized. To see the validity of this modification, notice that if $v = v'$ then $(\frac{v'}{v})^s = 1$ for all s , and the correct value is outputted. If $v \neq v'$, then the decryption algorithm outputs a uniformly distributed value in G , independent of m . The security of M-CS scheme against CCA2 is proven, based on the DDH assumption, in [5].

3 Public Key Broadcast Encryption Scheme

We use the definition in [9]. In a *public key broadcast encryption scheme* **BE**, a session key s is encrypted and broadcasted with the symmetric encryption of the “actual” message. Generally, the encryption of s is called the *enabling block*.

3.1 Public Key Broadcast Encryption Scheme

A *public key broadcast encryption scheme* **BE** consists of a 4-tuple of poly-time algorithms (**KeyGen**, **Reg**, **Enc**, **Dec**):

- **KeyGen**, the key generation algorithm, is a probabilistic algorithm used by the center to set up all the parameters of the scheme. **KeyGen** takes as input a security parameter 1^λ and a revocation threshold z (i.e. the maximum number of users that can be revoked) and generates the public key PK and the master secret key SK_{BE} .
- **Reg**, the registration algorithm, is a probabilistic algorithm used by the center to compute the secret initialization data needed to construct a new decoder each time a new user subscribes to the system. **Reg** receives as input the master secret key SK_{BE} and a (new) index i associated with the user; it returns the user’s secret key SK_i .
- **Enc**, the encryption algorithm, is a probabilistic algorithm used to encapsulate a given session key s within an enabling block T . **Enc** takes as input the public key PK , the session key s , and a set R of revoked users (with $|R| \leq z$) and returns the enabling block T .
- **Dec**, the decryption algorithm, is a deterministic algorithm that takes as input the secret key SK_i of user i and the enabling block T and returns the session key s that was encapsulated within T if i was a legitimate user when T was constructed, or the special symbol “ \perp ”.

3.2 Security against Adaptive Chosen Ciphertext Attack

An adversary \mathcal{A} in an *adaptive chosen ciphertext attack* (CCA2) is a probabilistic, poly-time *oracle query* machine. The attack game is defined in terms

of an interactive computation between the adversary and its environment. We describe the attack game used to define the security against CCA2; that is, we define the environment in which \mathcal{A} runs. We assume that the input to \mathcal{A} is 1^λ for some λ .

Stage 1: The adversary queries a key generation oracle. The key generation oracle computes $(PK, SK_{BE}) \leftarrow \mathbf{BE.KeyGen}(1^\lambda, z)$ and responds with PK .

Stage 2: The adversary enters the *user corruption stage*, where she is given oracle access to the *User Corruption Oracle* $Cor_{SK_{BE}}(\cdot)$. This oracle receives as input the index i of the user to be corrupted, computes $SK_i \leftarrow \mathbf{BE.Reg}(SK_{BE}, i)$ and returns the user's secret key SK_i . This oracle can be called adaptively for at most z times. Let us say that at the end of this stage the set R of at most z users is corrupted.

Stage 3: The adversary submits two session keys s_0, s_1 to an encryption oracle. On input s_0, s_1 , the encryption oracle computes: $\sigma \xleftarrow{R} \{0, 1\}$; $T^* \leftarrow \mathbf{BE.Enc}(PK, s_\sigma, R)$ and responds with the “target” enabling block T^* .

Stage 4: The adversary continues to make calls to the decryption oracle, subject only to the restriction that a submitted enabling block T is not identical to T^* .

Stage 5: The adversary outputs $\sigma^* \in \{0, 1\}$.

We define the advantage of \mathcal{A} as $Adv_{BE, \mathcal{A}}^{CCA2}(\lambda) = |Pr(\sigma^* = \sigma) - \frac{1}{2}|$

We consider a variant of the CCA2, *generalized chosen ciphertext attack* ($gCCA2$) [1,9]. The attack game of $gCCA2$ is the same as that of CCA2 except **Stage 4**. In the attack game of $gCCA2$, the adversary cannot ask about enabling blocks *closely related* to the “target” enabling block. That is, in **Stage 4**, the decryption oracle first checks whether *equivalence relation* $R_i(T, T^*)$ holds. If so, it outputs “ \perp ”.

Definition 1 (z-resilience of a public key broadcast encryption scheme)

We say that a public key broadcast encryption scheme \mathbf{BE} is z -resilient against CCA2 attack if for all probabilistic, poly-time oracle query machines \mathcal{A} , the function $Adv_{BE, \mathcal{A}}^{CCA2}(\lambda)$ grows negligibly in λ .

4 The DF Schemes

Y. Dodis and N. Fazio proposed three broadcast encryption schemes (we call them DF-CPA, DF- $gCCA$, DF-CCA2) that achieve z -resilience in an adaptive setting for the case of CPA (chosen plaintext attack), $gCCA2$, and CCA2, respectively. Subsequent schemes build on the previous one in an incremental manner. Therefore, the DF-CPA scheme is more efficient than the DF- $gCCA2$

Encryption algorithm Enc (PK, s, R)	Decryption algorithm Dec (i, T)
$E_1. r_1 \leftarrow_r Z_q$	$D_1. \alpha \leftarrow H(S, u_1, u_2, (j_1, H_{j_1}), \dots, (j_z, H_{j_z}))$
$E_2. u_1 \leftarrow g_1^{r_1}$	$D_2. \bar{v}_i \leftarrow u_1^{X_1(i)+Y_1(i)\alpha} \cdot u_2^{X_2(i)+Y_2(i)\alpha}$
$E_3. u_2 \leftarrow g_2^{r_1}$	$D_3. v_i \leftarrow EXP-LI(0, \dots, z; v_0, \dots, v_z)(i)$
$E_4. H_t \leftarrow h_t^{r_1}, (t = 0, \dots, z)$	$D_4. \text{if } v_i = \bar{v}_i$
$E_5. H_{j_t} \leftarrow EXP-LI(0, \dots, z; H_0, \dots, H_z)(j_t)$ $(t = 1, \dots, z)$	$D_5. \text{then } H_t \leftarrow u_1^{Z_1(i)} \cdot u_2^{Z_2(i)}$
$E_6. S \leftarrow s \cdot H_0$	$D_6. s \leftarrow \frac{S}{EXP-LI(j_1, \dots, j_z, i; H_{j_1}, \dots, H_{j_z}, H_i)(0)}$
$E_7. \alpha \leftarrow H(S, u_1, u_2, (j_1, H_{j_1}), \dots, (j_z, H_{j_z}))$	$D_7. \text{return } s$
$E_8. v_t \leftarrow c_t^{r_1} d_t^{r_1 \alpha}, (t = 0, \dots, z)$	$D_8. \text{else return } \perp$
$E_9. T \leftarrow \langle S, u_1, u_2, (j_1, H_{j_1}), \dots, (j_z, H_{j_z}), v_0, \dots, v_z \rangle$	

Fig. 1. DF-gCCA2

scheme and DF-gCCA2 scheme is more efficient than the DF-CCA2 scheme in the length of the enabling block and the computational overhead. In the next section, we define DF-gCCA2 and DF-CCA2. For a more detailed description, see [9].

4.1 DF-gCCA2

Key generation algorithm: KeyGen selects two random generators $g_1, g_2 \in G$, where G is a group of order q , in which q is a large prime such that $2q = p - 1$, and p is a large prime. **KeyGen** selects six z -degree polynomials $X_1(\xi), X_2(\xi), Y_1(\xi), Y_2(\xi), Z_1(\xi), Z_2(\xi)$ over Z_q , and computes $c_t = g_1^{X_1(t)} \cdot g_2^{X_2(t)}$, $d_t = g_1^{Y_1(t)} \cdot g_2^{Y_2(t)}$, $h_t = g_1^{Z_1(t)} \cdot g_2^{Z_2(t)}$, for $0 \leq t \leq z$. Finally, **KeyGen** chooses a hash function H from a family of \mathcal{F} of collision resistant hash functions, and outputs (PK, SK_{BE}) , where $PK = (p, q, g_1, g_2, c_0, \dots, c_z, d_0, \dots, d_z, h_0, \dots, h_z, H)$ and $SK_{BE} = (X_1, X_2, Y_1, Y_2, Z_1, Z_2)$.

Registration algorithm: Each time a new user $i > z$ decides to subscribe to the system, the center provides him with a decoder box containing the secret key $SK_i = (i, X_1(i), X_2(i), Y_1(i), Y_2(i), Z_1(i), Z_2(i))$.

Encryption algorithm: Using the ideas of the CS scheme [7,8], in order to obtain a non-malleable ciphertext, they “tag” each encrypted message so that it can be verified before proceeding with the actual decryption. In the broadcast encryption scenario, where each user has a different decryption key, the tag cannot be a single point - they need to distribute an entire EXP -polynomial $V(x)$. This is accomplished by appending $z+1$ tags, v_0, \dots, v_z , to the ciphertext.

The encryption algorithm receives as input the public key PK , the session key s , and a set $R = \{j_1, \dots, j_z\}$ of revoked users. It proceeds as described in Fig. 1, and finally it outputs T .

Decryption algorithm: To recover the session key, a legitimate user i can proceed as in Fig. 1. He computes the tag \bar{v}_i using his private key and then verifies the validity of the ciphertext by checking the interpolation of the $z+1$ values in point i against its \bar{v}_i (Step D_2, D_3 , and D_4). If i is a revoked user,

the algorithm fails in Step D_6 , since the interpolation points j_1, \dots, j_z, i are not pairwise distinct.

Security: The adversary can make the ciphertext malleable because of the use of an *EXP*-polynomial $V(x)$. Since each user i can verify the value of $V(x)$ in only one point, the adversary can modify v_0, \dots, v_z and construct a different *EXP*-polynomial $V'(x)$ such that $V'(x = x_i) = V(x_i)$, thus fooling user i to accept as valid a corrupted ciphertext. To prevent this, a family of *equivalence relations* $\{R_i\}$ is introduced. Two ciphertext T and T' are *equivalent* for user i if they have the same “data” components, and the tag “relevant to user i ” is correctly verified, i.e. $v_i = v'_i$ (even though other “irrelevant” tags could be different)[9]. By using this equivalent relation, DF-*gCCA2* is secure against *gCCA2*. In **Stage 4** of the attack game, the adversary cannot ask T which is *equivalently related* to the “target” T^* .

4.2 DF-CCA2

In Section 4.1, we saw that the DF-*gCCA2* scheme does not provide a complete solution to the CCA2 problem, but only suffices for *gCCA2* security. Indeed, given a challenge T^* with tag sequence v_0, \dots, v_z , it is trivial to make a different sequence v'_0, \dots, v'_z such that $v_i = v'_i$, resulting in a “different” enabling block $T \neq T^*$: however, $\text{Dec}(i, T^*) = \text{Dec}(i, T)$, allowing the adversary to “break” CCA2 security.

To achieve CCA2 security Dodis and Fazio used a trick to make the tag sequence v_0, \dots, v_z non-malleable. To this end, they used a *message authentication code* (MAC). The key generation algorithm and the registration algorithm are the same as those of DF-*gCCA2*. The encryption and decryption algorithm are shown in Fig. 2. The encryption algorithm operates similarly to the *gCCA2* encryption algorithm, but the main difference is that now a MAC key k is used to MAC the tag sequence v_0, \dots, v_z , and is encapsulated within T along with the session key s .

If the DDH problem is hard in G , H is chosen from a collision-resistant hash function family \mathcal{F} , and MAC is a *one-time* message authentication code, then the DF-CCA2 scheme is z -resilient against CCA2[9].

5 Proposed Scheme

In this section, we propose a new public key trace and revoke scheme secure against CCA2. Our scheme does not use the additional *one-time* MAC, so its security does not depend on the *one-time* MAC. The length of the enabling block of our scheme is about half that of the DF-CCA2 (DF-*gCCA2*) scheme. Additionally, the computational overhead of the user is lower than that of the DF-CCA2 (DF-*gCCA2*) scheme. Instead, the computational overhead of the server is increased, but the total computational overhead of the user and the server is the same as that of the DF-CCA2 (DF-*gCCA2*) scheme. We only consider the computation of exponentiation computed by the server and user. Our scheme is more

Encryption algorithm Enc (PK, s, R)	Decryption algorithm Dec (i, T)
$E_1. r_1 \leftarrow_r Z_q$	$D_1. \alpha \leftarrow H(S, u_1, u_2, (j_1, H_{j_1}), \dots, (j_z, H_{j_z}))$
$E_2. u_1 \leftarrow g^{r_1}$	$D_2. \bar{v}_i \leftarrow u_1^{X_1(i)+Y_1(i)\alpha} \cdot u_2^{X_2(i)+Y_2(i)\alpha}$
$E_3. u_2 \leftarrow g^{r_1}$	$D_3. v_i \leftarrow EXP-LI(0, \dots, z; v_0, \dots, v_z)(i)$
$E_4. H_t \leftarrow h_t^{r_1}, (t = 0, \dots, z)$	$D_4. \text{ if } v_i = \bar{v}_i$
$E_5. H_{j_t} \leftarrow EXP-LI(0, \dots, z; H_0, \dots, H_z)(j_t)$ $(t = 1, \dots, z)$	$D_5. \text{ then } H_t \leftarrow u_1^{Z_1(i)} \cdot u_2^{Z_2(i)}$
$E_6. k \leftarrow_r K$	$D_6. s k \leftarrow \frac{S}{EXP-LI(j_1, \dots, j_z, i; H_{j_1}, \dots, H_{j_z}, H_i)(0)}$
$E_7. S \leftarrow (s k) \cdot H_0$	$D_7. \text{ extract } s \text{ and } k \text{ from } (s k)$
$E_8. \alpha \leftarrow H(S, u_1, u_2, (j_1, H_{j_1}), \dots, (j_z, H_{j_z}))$	$D_8. \text{ if } \tau \neq MAC_k(v_0, \dots, v_z)$
$E_9. v_t \leftarrow c_t^{r_1} d_t^{r_1 \alpha}, (t = 0, \dots, z)$	$D_9. \text{ then return } \perp$
$E_{10}. \tau \leftarrow MAC_k(v_0, \dots, v_z)$	$D_{10}. \text{ else return } s$
$E_{11}. T \leftarrow \langle S, u_1, u_2, (j_1, H_{j_1}), \dots, (j_z, H_{j_z}), v_0, \dots, v_z, \tau \rangle$	$D_{11}. \text{ else return } \perp$

Fig. 2. DF-CCA2

efficient precisely because it does not require the computational overhead for the MAC but the DF-CCA2 scheme does. Our scheme is more practical, since the computing power of the user is weaker than the server in many applications.

Main Idea: In the DF-CCA2 scheme, given the enabling block $T \leftarrow \langle S, u_1, u_2, (j_1, H_{j_1}), \dots, (j_z, H_{j_z}), v_0, \dots, v_z, \tau \rangle$, to check the validity of T user i constructs $V(x)$ using v_0, \dots, v_z and checks whether $V(x = i) = v_i$. He also checks the validity of v_0, \dots, v_z by use of the MAC value τ . Our idea starts from the problem of the DF-gCCA2 scheme. In the DF-gCCA2 scheme, the decryption oracle cannot distinguish $V'(x)$ such that $V'(i) = V(i)$, but $v'_0, \dots, v'_z \neq v_0, \dots, v_z$. The DF-CCA2 scheme solves this problem by the use of the MAC.

We make the enabling block $T \leftarrow \langle S, u_1, u_2, c^r d^{r\alpha}, v_1, \dots, v_z \rangle$. Given T , user i computes $V(x)$ using v_1, \dots, v_z and his secret share v_i . Then he checks the validity of T using $c^r d^{r\alpha}$ and $V(x = 0)$. The adversary cannot compute $V(x = 0)$, since he knows only z shares of the degree- z polynomial $V(x)$. Therefore, the adversary cannot cheat the decryption oracle.

Key generation algorithm: **KeyGen** selects two random generators $g_1, g_2 \in G$, where G is a group of order q in which, q is a large prime such that $2q = p - 1$, and p is a large prime. It selects $x_1, x_2, y_1, y_2 \in Z_q$ and z -degree polynomials $X_1(\xi), X_2(\xi), Y_1(\xi), Y_2(\xi)$ over Z_q such that $X_1(0) = x_1, X_2(0) = x_2, Y_1(0) = y_1, Y_2(0) = y_2$. It also selects z -degree polynomials $Z_1(\xi), Z_2(\xi)$ over ξ and computes $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}$. Then, it computes $h_t = g_1^{Z_1(t)} g_2^{Z_2(t)}, 0 \leq t \leq z$ and $x_{1,t} = g_1^{X_1(t)}, x_{2,t} = g_2^{X_2(t)}, y_{1,t} = g_1^{Y_1(t)}, y_{2,t} = g_2^{Y_2(t)}, 0 \leq t \leq z$.

Finally, **KeyGen** chooses a hash function H from a family \mathcal{F} of collision resistant hash functions, and outputs (PK, SK_{BE}) , where $PK = (p, q, g_1, g_2, c, d, x_{1,0}, \dots, x_{1,z}, x_{2,0}, \dots, x_{2,z}, y_{1,0}, \dots, y_{1,z}, y_{2,0}, \dots, y_{2,z}, h_0, \dots, h_z, H)$ and $SK_{BE} = (X_1, X_2, Y_1, Y_2, Z_1, Z_2)$.

Registration algorithm: Each time a new user $i > z$ decides to subscribe to the system, the center provides him with a decoder box containing the secret key $SK_i = (i, X_1(i), X_2(i), Y_1(i), Y_2(i), Z_1(i), Z_2(i))$.

Encryption algorithm Enc (PK, s, R)	Decryption algorithm Dec (i, T)
$E_1. r_1 \leftarrow_r Z_q$ $E_2. u_1 \leftarrow g_1^{r_1}$ $E_3. u_2 \leftarrow g_2^{r_1}$ $E_4. H_t \leftarrow h_t^{r_1}, (t = 0, \dots, z)$ $E_5. H_{j_t} \leftarrow EXP-LI(0, \dots, z; H_0, \dots, H_z)(j_t)$ $(t = 1, \dots, z)$ $E_6. S \leftarrow s \cdot H_0$ $E_7. \alpha \leftarrow H(S, u_1, u_2)$ $E_8. C_t \leftarrow (x_{1,t} x_{2,t})^{r_1} (y_{1,t} y_{2,t})^{r_1 \alpha}$ $(t = 0, \dots, z)$ $E_9. C'_{j_t} \leftarrow EXP-LI(0, \dots, z; C_0, \dots, C_z)(j_t),$ $(t = 1, \dots, z)$ $E_{10}. C \leftarrow c^{r_1} d^{r_1 \alpha}$ $E_{11}. F_{j_t} = H_{j_t} \frac{C}{C'_{j_t}}, (t = 1, \dots, z)$ $E_{12}. T \leftarrow \langle S, u_1, u_2, c^{r_1} d^{r_1 \alpha},$ $(j_1, F_{j_1}), \dots, (j_z, F_{j_z}) \rangle$	$D_1. \alpha \leftarrow H(S, u_1, u_2)$ $D_2. C_i \leftarrow u_1^{X_1(i)+Y_1(i)\alpha} \cdot u_2^{X_2(i)+Y_2(i)\alpha}$ $D_3. H_i \leftarrow u_1^{Z_1(i)} \cdot u_2^{Z_2(i)}$ $D_4. F_i \leftarrow H_i \frac{C}{C_i}$ $D_5. s \leftarrow \frac{S}{EXP-LI(j_1, \dots, j_z, i; F_{j_1}, \dots, F_{j_z}, F_i)(0)}$

Fig. 3. Our Proposed scheme.

Encryption algorithm: Our scheme is based on the idea of M-CS [5]. The encryption algorithm receives as input the public key PK , the session key s , and a set $R = \{j_1, \dots, j_z\}$ of revoked users. It proceeds as described in Fig. 3, and finally it outputs T . **Enc** computes and distributes F_{j_t} , $1 \leq t \leq z$. We can think that $F_{j_t} = g_1^{Q(j_t)}$ where $Q(\xi)$ is z -degree polynomial in Z_q . Therefore, the adversary who only knows z shares of F_{j_t} cannot cheat the decryption oracle.

Decryption algorithm: To recover the session key, a legitimate user i can proceed as in Fig. 3. A legitimate user can compute s in Step D_5 , but the revoked user fails, since the interpolation of j_1, \dots, j_z, i are not pairwise distinct.

We here verify that the output of the decryption algorithm is identical to the session key s if the user i is a legitimate user. We can rewrite F_i computed from Step D_4 as follows (let $g_2 = g_1^w$):

$$\begin{aligned}
F_i &= H_i \cdot \left(\frac{C}{C_i} \right) \\
&= (u_1^{Z_1(i)} u_2^{Z_2(i)}) (c^{r_1} d^{r_1 \alpha}) (u_1^{-X_1(i)-Y_1(i)\alpha} \cdot u_2^{-X_2(i)-Y_2(i)\alpha}) \\
&= g_1^{r_1 Z_1(i) + w r_1 Z_2(i) - r_1 X_1(i) - r_1 Y_1(i)\alpha - w r_1 X_2(i) - w r_1 Y_2(i)\alpha} c^{r_1} d^{r_1 \alpha} \\
&= g_1^{r_1 Z_1(i) + w r_1 Z_2(i) - r_1 X_1(i) - r_1 Y_1(i)\alpha - w r_1 X_2(i) - w r_1 Y_2(i)\alpha + (r_1 x_1 + w r_1 x_2 + r_1 y_1 \alpha + w r_1 y_2 \alpha)} \\
&= g_1^{Q(i)} \\
&= g_1^{Q(i)}
\end{aligned}$$

Consequently, $F_i = g_1^{Q(i)}$ where $Q(\xi)$ is z -degree polynomial in Z_q . If we compute F_0 using the *Lagrange interpolation in the exponent* as in Step D_5 , we can obtain the following value:

$$\begin{aligned}
F_0 &= EXP-LI(j_1, \dots, j_z, i; F_{j_1}, \dots, F_{j_z}, F_i)(0) \\
&= g_1^{(r_1 z_1 + w r_1 z_2) - r_1 x_1 - r_1 y_1 \alpha - w r_1 x_2 - w r_1 y_2 \alpha + (r_1 x_1 + w r_1 x_2 + r_1 y_1 \alpha + w r_1 y_2 \alpha)} \\
&= H_0 \frac{c^{r_1} d^{r_1 \alpha}}{C} \\
&= H_0
\end{aligned}$$

Therefore, $\frac{S}{F_0} = \frac{(s \cdot H_0)}{H_0} = s$.

Security:

Theorem 1 *If the DDH problem is hard in G and H is chosen from a collision-resistant hash function family \mathcal{F} , then our scheme is z -resilient against the adaptive chosen ciphertext attack.*

Proof. Our overall strategy for the proof follows the structural approach in [8]. We shall define a sequence $\mathbf{G}_0, \mathbf{G}_1, \dots, \mathbf{G}_l$ of modified attack games. Each of the games $\mathbf{G}_0, \mathbf{G}_1, \dots, \mathbf{G}_l$ operates on the same underlying probability space. In particular, the public key cryptosystem, the coin tosses **Coins** of \mathcal{A} , and the hidden bit σ take on identical values across all games, while some of the rules that define how the environment responds to oracle queries may differ from game to game. For any $1 \leq i \leq l$, we let T_i be the event that $\sigma = \sigma^*$ in the game \mathbf{G}_i . Our strategy is to show that for $1 \leq i \leq l$, the quantity $|Pr[T_{i-1}] - Pr[T_i]|$ is negligible. In addition, it will be evident from the definition of game \mathbf{G}_1 that $Pr[T_1] = \frac{1}{2}$, which will imply that $|Pr[T_0] - \frac{1}{2}|$ is negligible.

Before continuing, we state the following simple but useful lemma in [8].

Lemma 1 *Let U_1, U_2 , and F be the events defined on some probability space. Suppose that the event $U_1 \wedge \neg F$ occurs if and only if $U_2 \wedge \neg F$ occurs. Then $|Pr[U_1] - Pr[U_2]| \leq Pr[F]$.*

Game \mathbf{G}_0 : Let \mathbf{G}_0 be the original attack game, let $\sigma^* \in \{0, 1\}$ denote the output of \mathcal{A} , and let T_0 be the event that $\sigma = \sigma^*$ in \mathbf{G}_0 , so that $Adv_{Ourscheme, \mathcal{A}}^{CCA2}(\lambda) = |Pr[T_0] - \frac{1}{2}|$.

Game \mathbf{G}_1 : \mathbf{G}_1 is identical to \mathbf{G}_0 , except that in \mathbf{G}_1 , steps E_4 and E_8 are replaced with the following:

$$\begin{aligned} E'_4. H_t &\leftarrow u_1^{Z_1(t)} \cdot u_2^{Z_2(t)}, t = 0, \dots, z \\ E'_8. C_t &\leftarrow u_1^{X_1(t)+Y_1(t)\alpha} \cdot u_2^{X_2(t)+Y_2(t)\alpha}, t = 0, \dots, z \end{aligned}$$

The change we have made is purely conceptual, it is just to make explicit any functional dependency of the above quantities on u_1 and u_2 . Clearly, it holds that $Pr[T_0] = Pr[T_1]$.

Game \mathbf{G}_2 : We again modify the encryption oracle, replacing steps E_1 and E_3 by

$$\begin{aligned} E'_1. r_1 &\leftarrow_r Z_q, r_2 \leftarrow_r Z_q \setminus \{r_1\} \\ E'_3. u_2 &\leftarrow g_2^{r_2} \end{aligned}$$

Notice that while in \mathbf{G}_1 the values u_1 and u_2 are obtained using the same value r_1 , in \mathbf{G}_2 they are independent subject to $r_1 \neq r_2$. Therefore, any difference in behavior between \mathbf{G}_1 and \mathbf{G}_2 immediately yields a PPT algorithm \mathcal{A}_1 that is able to distinguish DH tuples from totally random tuples with a non negligible advantage. That is, $|Pr[T_2] - Pr[T_1]| \leq \epsilon_1$ for some negligible ϵ_1 .

Game \mathbf{G}_3 : In this game, we modify the decryption oracle in \mathbf{G}_2 to obtain \mathbf{G}_3 as follows:

$$\begin{aligned}
D_1. \quad & \alpha \leftarrow H(S, u_1, u_2) \\
D'_2. \quad & C_i \leftarrow u_1^{X_1(i)+Y_1(i)\alpha+(X_2(i)+Y_2(i)\alpha)w} \\
D_{2-1}. \quad & \text{if } (u_2 = u_1^w) \\
D'_3. \quad & \text{then } H_i \leftarrow u_1^{Z_1(i)+Z_2(i)w} \\
D'_4. \quad & F_i \leftarrow H_i \frac{C}{C_i} \\
D'_5. \quad & s \leftarrow \frac{S}{\text{EXP-LI}(j_1, \dots, j_z, i, F_{j_1}, \dots, F_{j_z}, F_i)(0)} \\
D'_6. \quad & \text{else return } \perp
\end{aligned}$$

At this point, let R_3 be the event that the adversary \mathcal{A} submits some decryption queries that are rejected in Step D_{2-1} in \mathbf{G}_3 , but passed in \mathbf{G}_2 . Note that if a query passes in D_{2-1} in \mathbf{G}_3 , it would have also passed in \mathbf{G}_2 . It is clear that \mathbf{G}_2 and \mathbf{G}_3 proceed identically until the event R_3 occurs. In particular, the event $T_2 \wedge \neg R_3$ and $T_3 \wedge \neg R_3$ are identical. Therefore, by Lemma 1, we have

$$|Pr[T_3] - Pr[T_2]| \leq Pr[R_3]$$

and so it suffices to bound $Pr[R_3]$. To do this we consider two more games, \mathbf{G}_4 and \mathbf{G}_5

Game \mathbf{G}_4 : This game is identical to \mathbf{G}_3 , except for a change in Step E_6 as follows:

$$E'_6.e \leftarrow_r Z_q, S \leftarrow g_1^e$$

It is clear by construction that $Pr[T_4] = \frac{1}{2}$, since in \mathbf{G}_4 , the variable σ is never used at all, and so the adversary's output is independent of σ .

Let R_4 be the event that some decryption queries that would have passed in \mathbf{G}_2 , fail to pass in Step D_{2-1} in \mathbf{G}_4 . Then we have the following facts.

Lemma 2 $Pr[T_4] = Pr[T_3]$ and $Pr[R_4] = Pr[R_3]$.

The proof of Lemma 2 is shown in the Appendix

Game \mathbf{G}_5 : This game is identical to \mathbf{G}_4 , except for the following modification. In the decryption algorithm, we add the following *special rejection rule*, to prevent \mathcal{A} from submitting an illegal enabling block to the decryption oracle once she has received her challenge T^* .

Special rejection rule: After the adversary \mathcal{A} receives the challenge $T^* = (S^*, u_1^*, u_2^*, (c^r d^{r\alpha})^*, (j_1^*, F_{j_1}^*), \dots, (j_z^*, F_{j_z}^*))$, the decryption oracle rejects any query $< i, T >$, with $T = (S, u_1, u_2, (c^r d^{r\alpha}), (j_1, F_{j_1}), \dots, (j_z, F_{j_z}))$, such that $(S^*, u_1^*, u_2^*) \neq (S, u_1, u_2)$, but $\alpha = \alpha^*$, and it does so before executing the test in Step D_{2-1} .

To analyze this game, we define two events. Let C_5 be the event that the adversary \mathcal{A} submits a decryption query that is rejected using the above special rejection rule, and R_5 the event that the adversary \mathcal{A} submits some decryption query that would have passed in \mathbf{G}_2 , but fails to pass in Step D_{2-1} in \mathbf{G}_5 . Now it is clear that \mathbf{G}_4 and \mathbf{G}_5 proceed identically until event C_5 occurs. In particular, the event $R_4 \wedge \neg C_5$ and $R_5 \wedge \neg C_5$ are identical. Therefore, by Lemma 1, we have

$$|Pr[R_5] - Pr[R_4]| \leq Pr[C_5]$$

Now, if event C_5 occurs with non-negligible probability, we can construct a PPT algorithm \mathcal{A}_2 that breaks the collision resistance assumption with non-negligible probability. So, $|Pr[C_5]| \leq \epsilon_2$ for some negligible ϵ_2 .

Finally, we show that event R_5 occurs with negligible probability.

Lemma 3 $Pr[R_5] \leq \frac{Q_{\mathcal{A}}(\lambda)}{q}$.

Where, $Q_{\mathcal{A}}(\lambda)$ is an upper bound on the number of decryption queries made by the adversary \mathcal{A} . The proof of Lemma 3 is shown in the Appendix.

Finally, combining the intermediate results, we conclude that the adversary \mathcal{A} 's advantage is negligible:

$$Adv_{OurScheme, \mathcal{A}}^{CCA2}(\lambda) \leq \epsilon_1 + \epsilon_2 + \frac{Q_{\mathcal{A}}(\lambda)}{q} \quad \square$$

References

1. J.H. An, Y. Dodis, T. Rabin, On the security of joint signature and encryption, *EUROCRYPT'02, LNCS V.2332*, pp.83-107, 2002.
2. T. Asano, A revocation scheme with minimal storage at receivers, *ASIACRYPT'02, LNCS V.2501*, pp.433-450, 2002.
3. D. Boneh, M. Franklin, An efficient public key traitor tracing scheme, *CRYPTO'99, LNCS V.1666*, pp.338-353, 1999.
4. D. Boneh, J. Shaw, Collusion-secure fingerprinting for digital data, *IEEE Transaction on Information Theory* 44(5), pp.1897-1905, 1998.
5. R. Canetti, S. Goldwasser, An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack, *EUROCRYPT'99, LNCS V.1592*, pp.90-106, 1999.
6. B. Chor, A. Fiat, M. Naor, Tracing traitor, *CRYPTO'94, LNCS V.839*, pp.257-270, 1994.
7. R. Cramer, V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, *CRYPTO'98, LNCS V.1462*, pp.13-25, 1998.
8. R. Cramer, V. Shoup, Design and analysis of practical public key encryption scheme secure against adaptive chosen ciphertext attack, *Manuscript*, 2001.
9. Y. Dodis, N. Fazio, Public key trace and revoke scheme secure against adaptive chosen ciphertext attack, *PKC'03*, pp.100-115, 2003.
10. Y. Dodis, N. Fazio, Public key trace and revoke scheme secure against adaptive chosen ciphertext attack, *Full version of [9]*
11. A. Fiat, M. Naor, Broadcast encryption, *CRYPTO'93, LNCS V.773* pp.480-491, 1993.
12. E. Gafni, J. Staddon, Y.L. Yin, Efficient methods for integrating traceability and broadcast encryption, *CRYPTO'99, LNCS V.1666*, pp.372-287, 1999.
13. D. Halevy, A. Shamir, The LSD broadcast encryption scheme, *CRYPTO'02, LNCS, V.2442*, pp.47-60, 2002.
14. K. Kurosawa, Y. Desmedt, Optimum traitor tracing and asymmetric schemes, *EUROCRYPT'98, LNCS V.1403*, pp.145-157, 1998.
15. D. Naor, M. Naor, J. Lostpiech, Revocation and tracing schemes for stateless receivers, *CRYPTO'01, LNCS V.2139*, pp.41-62, 2001.
16. M. Naor, B. Pinkas, Threshold traitor tracing, *CRYPTO'98, LNCS V.1462*, pp.502-517, 1998.

17. B. Pfitzmann, Trials of traced traitors, *Workshop on Information Hiding, LNCS V.1174*, pp.49-64, 1996.
18. B. Pfitzmann, M. Waidner, Asymmetric fingerprinting for large collusions, *ACM conference on Computer and Communication Security*, pp.151-160, 1997.
19. D.R. Stinson, R. Wei, Combinatorial properties and constructions of traceability schemes and frameproof codes, *SIAM Journal on Discrete Math* 11(1), pp.41-53, 1998.
20. W.G. Tzeng, Z.J. Tzeng, A public-key tracing scheme with revocation using dynamic shares, *PKC'01, LNCS 1992*, pp.207-224, 2001.
21. D.M. Wallner, E.J. Harder, R.C. Agee, Key management for multicast: Issues and Architectures, *IETF Network Working Group, RFC 2627*, 1999.
22. C.K. Wong, M. Gouda, S. Lam, Secure group communications using key graphs, *ACM SIGCOMM'98*, pp.68-79, 1998.

Appendix

To prove Lemma 2 and Lemma 3, the following lemma is useful. The proof of Lemma 4 is shown in [8]. Our proofs follow the structural approach in [8,10]. Therefore, they are similar to that of [10] except for some variables and notations.

Lemma 4 *Let k, n be integers with $1 \leq k \leq n$, and let K be a finite field. Consider a probability space with random variables $\alpha \in K^{n \times 1}$, $\beta = (\beta_1, \dots, \beta_k)^T \in K^{k \times 1}$, $\gamma \in K^{k \times 1}$, and $M \in K^{k \times n}$, such that α is uniformly distributed over $K^{n \times 1}$, $\beta = M\alpha + \gamma$, and for $1 \leq i \leq k$, the i th rows of M and γ are determined by $\beta_1, \dots, \beta_{i-1}$.*

Then conditioning on any fixed values of $\beta_1, \dots, \beta_{k-1}$ such that the resulting matrix M has rank k , the value of β_k is uniformly distributed over K in the resulting conditional probability space.

In what follows, we define:

Coins: the coin tosses of \mathcal{A} ; $X_t = X_1(t) + wX_2(t)$, $Y_t = Y_1(t) + wY_2(t)$, $Z_t = Z_1(t) + wZ_2(t)$, $t = 0, \dots, z$;
 $w = \log_{g_1} g_2$

Proof of Lemma 2

Lemma 2. $Pr[T_4] = Pr[T_3]$ and $Pr[R_4] = Pr[R_3]$.

Proof. Consider the quantity $X := (\text{Coins}, H, w, X_1(0), \dots, X_1(z), X_2(0), \dots, X_2(z), Y_1(0), \dots, Y_1(z), Y_2(0), \dots, Y_2(z), Z_1, \dots, Z_z, \sigma, r_1^*, r_2^*)$ and the quantity Z_0 . Note that X and Z_0 take on the same values in \mathbf{G}_3 and \mathbf{G}_4 . Consider also the quantity $e^* = \log_{g_1} S^*$. This quantity takes on different values in \mathbf{G}_3 and \mathbf{G}_4 . For clarity, let us denote these values as $[e^*]_3$ and $[e^*]_4$, respectively.

It is clear by inspection that the events R_3 and T_3 are determined as functions of X , Z_0 , and $[e^*]_3$. Also, the events R_4 and T_4 are determined as functions of X , Z_0 and $[e^*]_4$. Therefore to prove Lemma 2, it suffices to show that the distributions of $(X, Z_0, [e^*]_3)$ and $(X, Z_0, [e^*]_4)$ are identical. Observe that by the construction, conditioning on any fixed values of X and Z_0 , the distribution of $[e^*]_4$ is uniform over Z_q . Therefore, it will suffice to show that conditioning on any fixed values of X and Z_0 , the distribution of $[e^*]_3$ is uniform over Z_q .

We have the following equation:

$$\begin{pmatrix} Z_0 \\ [e^*]_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w \\ r_1^* & wr_2^* \end{pmatrix}}_M \cdot \begin{pmatrix} Z_1(0) \\ Z_2(0) \end{pmatrix} + \begin{pmatrix} 0 \\ \log_{g_1} s_\sigma \end{pmatrix}$$

where $\det(M) = w(r_2^* - r_1^*) \neq 0$ since $r_2^* \neq r_1^*$.

Conditioning only on a fixed value of X , the matrix M is fixed, but the values $Z_1(0)$ and $Z_2(0)$ are still uniformly and independently distributed over Z_q . If we further condition on a fixed value of Z_0 , the value of s_σ is fixed; hence, by Lemma 4, the distribution of $[e^*]_3$ is uniform over Z_q . \square

Proof of Lemma 3

Lemma 3. $Pr[R_5] \leq \frac{Q_A(\lambda)}{q}$.

Proof. For $1 \leq j \leq Q_A(\lambda)$, we define the following events;

- $R_5^{(j)}$: the event that the j th ciphertext $\langle i, T \rangle$, submitted to the decryption oracle in \mathbf{G}_5 , fails to pass D_{2-1} , but would have passed in \mathbf{G}_2 ,
- $B_5^{(j)}$: the event that the j th ciphertext $\langle i, T \rangle$, submitted to the decryption oracle *before* \mathcal{A} received her challenge,
- $\hat{B}_5^{(j)}$: the event that the j th ciphertext $\langle i, T \rangle$, submitted to the decryption oracle *after* \mathcal{A} received her challenge.

If we show that $Pr[R_5^{(j)} | B_5^{(j)}] \leq \frac{1}{q}$ and $Pr[R_5^{(j)} | \hat{B}_5^{(j)}] \leq \frac{1}{q}$, then Lemma 3 is proved. \square

Lemma 5 For all $1 \leq j \leq Q_A(\lambda)$, we have $Pr[R_5^{(j)} | B_5^{(j)}] \leq \frac{1}{q}$.

Lemma 6 For all $1 \leq j \leq Q_A(\lambda)$, we have $Pr[R_5^{(j)} | \hat{B}_5^{(j)}] \leq \frac{1}{q}$.

Proof of the Lemma 5. Fix $1 \leq j \leq Q_A(\lambda)$ and consider the quantities:

$$X := (Coins, H, w, Z_0, \dots, Z_z), X' := (X_0, \dots, X_z, Y_0, \dots, Y_z)$$

These two quantities completely determine the behavior of the adversary up to the moment that \mathcal{A} performs the encryption query, and in particular, they completely determine the event $B_5^{(j)}$. Let us call X and X' *relevant* if the event $B_5^{(j)}$ occurs. Hence to prove Lemma 5, it suffice to prove that the probability of event $R_5^{(j)}$, conditioned on any *relevant* values of X and X' , is less than $\frac{1}{q}$.

The test D_{2-1} fails if and only if $u_2 \neq u_1^w$. Thus if the test in D_{2-1} fails but would have passed in G_2 , it must be the case that $u_2 \neq u_1^w$ and $c^{r_1} d^{r_1 \alpha} = EXP-LI(j_1, \dots, j_z, i: C_{j_1}, \dots, C_{j_z}, C_i)(0)$. Taking the logs (base g_1), the condition $u_2 \neq u_1^w$ is equivalent to $r_2 \neq r_1$. If we let $\beta = \log_{g_1} c^{r_1} d^{r_1 \alpha}$ and $\hat{\beta} = \log_{g_1} EXP-LI(j_1, \dots, j_z, i: C_{j_1}, \dots, C_{j_z}, C_i)(0)$, then $\hat{\beta} = r_1 X_1(0) + w r_2 X_2(0) + \alpha r_1 Y_1(0) + \alpha w r_2 Y_2(0)$. Notice that $\hat{\beta}$ can be expressed in terms of $(X_1(0), X_2(0), \dots, X_1(z),$

$X_2(z), Y_1(0), Y_2(0), \dots, Y_1(z), Y_2(z))^T$. Therefore, we can make the following equation (for details, see [10]):

$$\begin{pmatrix} X_0 \\ \vdots \\ X_z \\ Y_0 \\ \vdots \\ Y_z \\ \hat{\beta} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & & & & & & & \vdots \\ 0 & 0 & \cdots & 1 & w & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & w & \cdots & 0 & 0 \\ \vdots & & & & & & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & w \\ \sigma_0 & \sigma_1 & \cdots & \sigma_{2z} & \sigma_{2z+1} & \sigma_{2z+2} & \sigma_{2z+3} & \cdots & \sigma_{4z+2} & \sigma_{4z+3} \end{pmatrix}}_M \cdot \begin{pmatrix} X_1(0) \\ X_2(0) \\ \vdots \\ X_1(z) \\ X_2(z) \\ Y_1(0) \\ Y_2(0) \\ \vdots \\ Y_1(z) \\ Y_2(z) \end{pmatrix}$$

Let us first fix X , which fixes the first $2z+2$ rows of the matrix M , but the values $(X_1(0), X_2(0), \dots, Y_1(z), Y_2(z))$ are still uniformly distributed over Z_q . Next fix X' such that X and X' are *relevant* and $r_1 \neq r_2$. Then the last row of the matrix M is fixed. From this, it follows by Lemma 4 that $\hat{\beta}$ is uniformly distributed over Z_q , but β is fixed, we have $\Pr[\beta = \hat{\beta}] = \frac{1}{q}$. \square

Proof of the Lemma 6. Fix $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$ and consider the quantities:

$$X := (\text{Coins}, H, w, Z_0, \dots, Z_z, r_1^*, r_2^*, e^*), \quad X' := (X_0, \dots, X_z, Y_0, \dots, Y_z, \beta^*).$$

where $\beta^* = \log_{g_1}(c^{r_1} d^{r_1 \alpha})^*$ and $i > z$. The values of X and X' completely determine the adversary's entire behavior in Game G_5 , in particular, they completely determine the event $\hat{B}_5^{(j)}$. Let us call X and X' *relevant* if the event $\hat{B}_5^{(j)}$ occurs. It will suffice to prove that conditioned on any fixed, *relevant* values of X and X' , the probability that $R_5^{(j)}$ occurs is bounded by $\frac{1}{q}$. As in the proof of Lemma 5, we have the following equation (for the detail, see [10]):

$$\begin{pmatrix} X_0 \\ \vdots \\ X_z \\ Y_0 \\ \vdots \\ Y_z \\ \beta^* \\ \hat{\beta} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & & & & & & & \vdots \\ 0 & 0 & \cdots & 1 & w & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & w & \cdots & 0 & 0 \\ \vdots & & & & & & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & w \\ \sigma_0^* & \sigma_1^* & \cdots & \sigma_{2z}^* & \sigma_{2z+1}^* & \sigma_{2z+2}^* & \sigma_{2z+3}^* & \cdots & \sigma_{4z+2}^* & \sigma_{4z+3}^* \\ \sigma_0 & \sigma_1 & \cdots & \sigma_{2z} & \sigma_{2z+1} & \sigma_{2z+2} & \sigma_{2z+3} & \cdots & \sigma_{4z+2} & \sigma_{4z+3} \end{pmatrix}}_M \cdot \begin{pmatrix} X_1(0) \\ X_2(0) \\ \vdots \\ X_1(z) \\ X_2(z) \\ Y_1(0) \\ Y_2(0) \\ \vdots \\ Y_1(z) \\ Y_2(z) \end{pmatrix}$$

Again conditioning on a fixed value of X and X' , we have that $\hat{\beta}$ is uniformly distributed over Z_q , but β^* is fixed. Therefore, we have $\Pr[\beta^* = \hat{\beta}] = \frac{1}{q}$. \square

Sequential Key Derivation Patterns for Broadcast Encryption and Key Predistribution Schemes

Nuttapong Attrapadung, Kazukuni Kobara, and Hideki Imai

Imai Laboratory, Institute of Industrial Science, University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan
nuts@iimailab.iis.u-tokyo.ac.jp, {kobara,imai}@iis.u-tokyo.ac.jp

Abstract. We study two closely related primitives: Broadcast Encryption and Key Predistribution Schemes (KPS). Broadcast Encryption allows a broadcaster to broadcast an encrypted message so that only a designated group of users can decrypt it. KPS allows a designated group of users to establish a common key non-interactively. We discover a generic method to construct efficient broadcast encryption schemes and KPSs naturally from Pseudo-Random Sequence Generators (PRSG) by observing that there are general “patterns” to do so. The two currently best PRSG-based broadcast encryption schemes such as the “Subset Difference” (SD) scheme by Naor Naor and Lotspiech and its refinement, the “Layered SD” (LSD) scheme by Halevy and Shamir, are indeed two special cases of our method. We demonstrate the power of this generic method by giving: (1) A solution to the most challenging variant of KPS: the one which supports arbitrary number of users to form a group yet secure against any collusion. We obtain a lower bound of the private key size at each user for any PRSG-based KPSs in this setting and construct a KPS that meets this bound. (2) An evidence that previous PRSG-based BE schemes, such as SD and LSD, can be further improved without any further assumption using this general method. We construct “Flexible SD” and “Flexible LSD” broadcast encryption schemes, which require less private key size while still maintain exactly the same broadcast size compared to their original SD/LSD schemes.

1 Introduction

Our main contribution is a generic method to construct efficient schemes of the two following closely related primitives naturally from Pseudo-Random Sequence Generators (PRSG). The primitives are:

Key Predistribution Scheme. Key Predistribution Scheme (KPS) involves n users. Each user is given a unique private key. For a group of users $P \subseteq N = \{1, \dots, n\}$, any users in P should be able to non-interactively compute a common key k_P using only its private key while other receivers outside P should not be able to do so even if they collude. Such a scheme is motivated by the

scenario of secure conferences over network. KPS can be viewed as a special case of broadcast encryption as we will see below.

Broadcast Encryption. Broadcast encryption (BE) involves 1 broadcaster and n receivers. Each receiver is given a unique private key. The broadcaster is given a broadcaster key. The broadcaster wishes to broadcast messages to a designated set $P \subseteq N = \{1, \dots, n\}$ of receivers. Any receivers in P should be able to decrypt the broadcast message using only its private key while other receivers outside P should not be able to do so even if they collude. A broadcast encryption scheme is sometimes called a *Revocation Scheme*, where one is interested in a subset of non-privileged users or so-called revoked users rather than privileged ones hence its name. Such a scheme is motivated largely by pay-TV systems, the distribution of copyrighted material.

Relating 2 Primitives. In BE, a body of message is typical long and should be encrypted by a key commonly known to P . We call such a key a message encryption key Mek. To share Mek among P the broadcaster produces a header Hdr such that given Hdr and a private key of user in P one can obtain Mek. If the private keys are generated by KPS, each user in P already has a common key before hand thus there is no need of Hdr in this case. In this sense, KPS is known as *zero-header* BE.

Overview on Previous Works. KPSs were introduced by Blom [5] and formalized by Matsumoto-Imai [20]. Broadcast encryption schemes were first formally studied by Fiat-Naor [13]. Since then, many variants of the basic problem of KPSs and BEs are proposed. The relations of two primitives are also captured in many works (see, e.g., [18]). Since a KPS can be viewed as a special case of a BE, each variant of KPSs will be a variant of BEs (but not the converse). Keep in mind that a KPS is a zero-header BE. Therefore it is enough to describe variants of BEs as follows. To name just a few, the scheme might support bounded or unbounded number of privileged users and/or the maximum number of adversarial coalition; the privileged subset of users can be fixed, slowly changing, rapidly changing; the keys stored by each user can be stateful or stateless (to be updated or not); it might be possible to trace a traitor who illegally leak its secret key in the scenario so-called *tracing-traitor*; the scheme might be symmetric-key or public-key; and so on. We found that it is convenient to categorize the relevant schemes by their approaches as follows. For BEs there are (1)combinatorial approaches: schemes using combinatorial design such as [6,23,19,18,17,15,4]; and schemes using tree structure such as [25,24,21,16,10,3] and (2)algebraic approaches: schemes using secret sharing scheme on the exponent to perform ElGamal-like encryption such as [2,22,11,12]. For KPSs almost all of them are using combinatorial approaches. Most of the past works for KPSs can be found in Kurosawa, et al. [18].

The Most Challenging Variant. We study the most challenging variant of BE (and KPS) where it supports unbounded number of users in privileged

subsets; unbounded number of revoked users allowed to form adversarial coalition (adaptively by central adversary); the privileged subset does not depend on the history; the private key stored by each user is stateless, i.e., it is fixed from the initialization time. This combined variant is arguably the hardest but the most desired one especially stateless scenario as argued explicitly first by Naor-Naor-Lotspiech [21].

The Main Goal and Some Solutions. The main goal towards BE and KPS problems is to construct efficient schemes satisfying the above mentioned variant where for KPS: the private key size is small, and for BE: both the header size and the private key size are small in the function of n , $|P|$, or $r := n - |P|$. A BE scheme which solves above mentioned variant problem and satisfies good efficiency in only one side is trivial. On one side, the private key size is independent of n but the header size is linear in $|P|$. On the other side, the header size is zero but the private key size is exponential in n . Note that the latter is a trivial KPS, which is definitely inefficient, however, is considered the best known solution for the above mention variant of KPS. As opposed to KPS, there are many BE schemes which have efficiency far better than the trivial schemes. One solution which is considered a ground work to many consequent works is due to Naor-Naor-Lotspiech [21]. It associates each user with the leaf of a balanced binary tree yielding a scheme called complete subtree (CS) in which the header size is $O(r \log(n/r))$ and the private key size is $O(\log n)$.

Major improvement to this idea were the subset difference (SD) method in their same paper [21] and its refinement, layered SD method, by Halevi-Shamir [16]. Both obtain the header size $O(r)$. While the SD scheme obtains the private key size $O(\log^2 n)$, the LSD scheme obtains the private key size $O(\log^{1+\epsilon} n)$ for small $\epsilon > 0$. More recent improvement due to Asano [3] utilizes the master key technique of Chick-Tavares [8] on balanced a -ary tree version of CS where $a > 2$ (instead of binary tree). This scheme obtains the header size $O(r(\log_a(n/r) + 1))$ and the private key size $O(1)$. These 3 schemes are considered the current state of the art for BE in the sense that while SD/LSD scheme obtain less header size, utilize a weak computational assumption, obtain much less computational cost; Asano's scheme obtains minimum private key size.

The basic idea of the schemes above is a mechanism called *subset-cover framework* [21]. Such a scheme in this framework varies to one another by (1) an underlying collection of subsets of a particular form, and (2) techniques which make use of computational assumption to enable the generation of many *computational unrelated* private keys. The improvements of recent works are primarily due to sophisticated design of the underlying collection in (1) to shorten the header size, and utilization of technique in (2) to shorten the private key size.

Shortening Private Key Size. Various methods to shorten the private key size are depicted in Figure 1. We capture these in 4 types. For simplicity, let us consider KPS where $N = \{1, 2, 3\}$. Each user u is supposed to be able to compute the common key k_S of set $S \subseteq N$ where $u \in S$. In the trivial KPS, user $u \in N$ just stores $\{k_S : u \in S \subseteq N\}$ as the private key set. The goal now

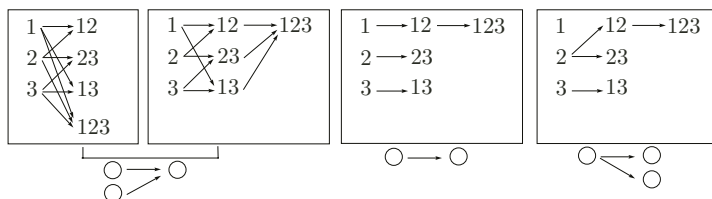


Fig. 1. Various methods to shorten key size: type 1-4 from left to right.

is to reduce the number of elements in the private key set (recall that KPS is zero-header BE thus we do not worry about reducing header for now).

A natural way to do so is to let each user keep one master key which can be used to derive all common keys that he is supposed to be able to compute. This method is shown as type 1 in Figure 1. We denote by $A \rightarrow B$ where $A \subset B \subseteq N$ a one-way computation which takes as input k_A outputs k_B , i.e., one can easily compute k_B given k_A but given k_B it is hard to compute k_A . Note that this arrow notation applies to all types in the figure. Observe that every method in type 2 in fact also falls into type 1. The good functionality of one-way computation of these first 2 types is that: for any different inputs which are intractable to compute given one another, one can design a one-way computation such that it results in the same output. This functionality give these first 2 types the very short private key as one master key: $k_{\{u\}}$ for each user u .

The master key trick is originally introduced by Akl-Taylor [1] and Chick-Tavares [8] and brought to the context of BE first by Asano [3]. This trick uses the RSA assumption and falls into type 2. Another elegant trick for mastering key is proposed recently by Boneh-Silverberg [7]. Their scheme utilizes multilinear forms and assumes the Diffie-Hellman inversion assumption. This trick falls into type 1. Now that we have BEs with zero-header and the private key size as minimum as possible, so does this mean that we are done? The answer is no. The reasons are as follows. For the trick using multilinear form, unfortunately there is *no* known concrete construction of such forms up to date and it is believed that it is hard to find ones as argued by the authors themselves. For the trick using RSA, it turns out that such a scheme requires a large number of primes as $O(2^n)$ which is extremely inefficient both for storing and for generation. Note that, however, the trick using RSA works fine for non-zero-header BEs [3]. Nonetheless, for the trick using RSA in all cases, a critical disadvantage besides the issue of primes is a heavy computation due to the modular exponentiation with the exponent being products of many primes without knowing the order.

Type 3 is implicitly mentioned first in Akl-Taylor [1]. In this type the functionality of one-way computation as opposed to the first 2 types allows only one input per one output. User u just stores k_B whenever $u \in B \subseteq N$ and $u \notin A$ where $A \rightarrow B$ and A appears just before B in the diagram. The method of type 3 just makes use of any length-preserving one-way functions. A natural way to generalize this idea is to use any one-way functions that expand the length of inputs such as pseudo-random sequence generator (PRSG) $G_d()$, $d \in \mathbb{Z}^+$ which d -ples the input, i.e., whose the bit length of the output is d times that of the

input so that from one key, d many keys can be derived. This idea falls into type 4. The schemes which *implicitly* use this type of method in the context of BE are the SD [21] and LSD [16] method. Note that there is no KPS constructed by using this technique before. Shortening the private key size using only PRSG in type 4 has an obvious advantage over the one using RSA in type 2 since it needs no prime and its computation is much more efficient. Moreover, and perhaps more critical, is that the existence of PRSG is considered a weak assumption. Basing a cryptographic system only on a weak assumption is always preferred.

Motivation. The question is how one can use the method of type 4, shortening the private key size by utilizing PRSG, *at its most beneficial*. We take a look back into the schemes which implicitly utilize this method: the SD and LSD schemes. Although the balance tree representation which is used in SD/LSD has good properties since it somehow captures the nature of PRSGs in the sense that it utilizes PRSG by letting some values related to some parent nodes input to a PRSG so that a PRSG outputs some values related to their child nodes. However, the strict structure of subset difference is too rigid to capture the good properties from PRSGs thus the optimality of LSD scheme which argued by Halevi-Shamir [16] in their work is worked *only for their structure*. To obtain the most beneficial from PRSG, more flexible generalization of the idea must be rigorously captured.

Our Main Results. We observe that there are general “patterns” to construct broadcast encryption schemes and KPSs naturally from PRSGs. We call such a pattern a **sequential key derivation pattern**. We demonstrate the power of this general patterns by giving:

1. A solution to the most challenging variant of KPS: the one which supports arbitrary number of users to form a group yet secure against any collusion. We obtain a lower bound of the private key size at each user for any PRSG-based KPSs in this setting. This lower bound of the private key size appears to be $O(1/n)$ times that of information-theoretically secure KPS in the same setting [9]. We then propose an optimal construction of KPS which meets the bound. This construction makes use of a new combinatorial structure which is of an independent interest.
2. An evidence showing that previous best PRSG-based BE schemes, SD and LSD, can be further improved without any further assumption by using this general method. To do this, we construct “Flexible SD”(FSD) and “Flexible LSD”(FLSD) broadcast encryption schemes. Such schemes require, although asymptotically the same, less *exact* private key size while still maintain exactly the same broadcast size compared to the original SD/LSD schemes. More concretely, the number of private keys are reduced at most $\log n$ from the original schemes. This reduction depends on the user index. In particular, in the FSD and FLSD scheme, there are exactly $n/2$ users and $n/2^{\sqrt{\log n}}$ users who store exactly $\log n$ fewer keys than that of the SD and (Basic) LSD scheme respectively.

Table 1. Summary of results compared to previous works. The parameter in each KPS is the storage size at user: the first term in the addition is private key size, the second term is non-secret storage. The parameters in each BE consists of the private key size (in terms of the exact number of elements) in the upper row, and the header size (in terms of bound on the exact number of elements) in the lower row. The parameter w_u, z_u are functions of user index u , for $u \in N$ (see Theorem 2 and 3 for detail).

Based on \rightarrow	Multilinear form (type 1)	RSA (type 2)	PRSG (type 4)
Generic utilization of technique	[7]	[8]	(This work)
KPS	[7] $O(1) + O(n)$	[3](implicitly), [4] $O(1) + O(2^n \log n)$	(This work) $O(\frac{2^n}{n}) + 0$
BE	-	[3] 1 $r(\log_a(\frac{n}{r}) + 1)$	SD[21] $\text{key}_{SD} = (\log^2 n + \log n)/2 + 1$ $2r - 1$ (Basic)LSD[16] $\text{key}_{LSD} = \log^{3/2} n + 1$ $4r - 2$ FSD(This work) $\text{key}_{SD} - w_u, 0 \leq w_u \leq \log n, u \in N$ $2r - 1$ FLSD(This work) $\text{key}_{LSD} - z_u, 0 \leq z_u \leq \log n, u \in N$ $4r - 2$

2 Definitions

Definition 1 (BROADCAST ENCRYPTION, BE). A *Broadcast Encryption Scheme* (BE) is a 3-tuple of polynomial-time algorithms (Keygen, Encrypt, Decrypt), where:

- BE.Keygen**($1^\lambda, n$): Takes as input a security parameter 1^λ , the number of users n . It outputs n sets of receiver keys I_1, \dots, I_n and a sender key T .
- BE.Encrypt**(P, T, Mek): Takes as input a subset $P \subseteq N := \{1, \dots, n\}$ of privileged users, the sender key T , and a message encryption key Mek . It outputs a header Hdr of the ciphertext.
- BE.Decrypt**(P, Hdr, I_u): Takes as input a subset $P \subseteq N$, a header Hdr , and a receiver key I_u . It outputs the message encryption key Mek that was sent if u was in the set P , or the special symbol \perp otherwise.

In practice, it is used in conjunction with a symmetric encryption algorithm F to encrypt the message body M under the message encryption key Mek resulting in a broadcast body C_M . The broadcast to receivers consists of (P, Hdr, C_M) .

The security notion for broadcast encryption that we concern here is the one considered by Naor, et al. [21], where the security against chosen-ciphertext attack with adaptive user-corruption is defined.

In the same paper [21], the authors presented the subset-cover algorithm as a sufficient condition to construct such a broadcast encryption scheme. Here we recap its definition as the following. Notice that it is slightly different in context from the original one.

Definition 2 (SUBSET-COVER ALGORITHM, SC) *A subset cover algorithm SC is a 2-tuple of polynomial time algorithms (DefineSet, Cover), where:*

- SC.DefineSet(n) :Takes as input the number of users n . It outputs a family \mathcal{S} of subsets of N and a user structure Γ (for example, a binary tree of users).
 SC.Cover(P, \mathcal{S}) :Takes as input a privileged subset P of users, the family \mathcal{S} defined from DefineSet. It outputs a partition $\mathcal{S}_P := \{S_{i_1}, S_{i_2}, \dots, S_{i_m} : S_{i_j} \in \mathcal{S}\}$ of P , i.e., $P = \bigsqcup_{j=1}^m S_{i_j}$, such that the number of subsets in its is the minimum among all possible partitions of P by \mathcal{S} .

A broadcast encryption in the subset-cover framework is a broadcast encryption scheme that makes use of subset-cover algorithm as its subalgorithm as follows.

- BE.Keygen($1^\lambda, n$) Run SC.DefineSet(n) to get (\mathcal{S}, Γ) . From Γ , it determines subset key $\mathbf{k}(S_i)$ for each $S_i \in \mathcal{S}$. Then it defines I_u to be the set of λ -bits strings containing the minimal elements yet still being sufficient to easily deduce each subset key $\mathbf{k}(S_i)$ where $u \in S_i$ from I_u . The broadcaster key T is the set consisting of all the subset keys.
 BE.Encrypt(P, T, Mek) Run SC.Cover(P, \mathcal{S}) to obtain $\{S_{i_1}, S_{i_2}, \dots, S_{i_m}\}$. The Mek is encrypted by an encryption scheme E by each subset key $\mathbf{k}(S_{i_j})$, $j = 1, \dots, m$ yielding a Hdr:

$$\langle (i_1, E_{\mathbf{k}(S_{i_1})}(\text{Mek})), \dots, (i_m, E_{\mathbf{k}(S_{i_m})}(\text{Mek})) \rangle$$

- BE.Decrypt(P, Hdr, I_u) Parse Hdr as $\langle (i_1, c_1), \dots, (i_m, c_m) \rangle$, it finds i_j such that $u \in S_{i_j}$ (in case $u \notin P$ the result is null). Denote D the decryption algorithm corresponding to E . It uses I_u to derive $\mathbf{k}(S_{i_j})$ then computes $D_{\mathbf{k}(S_{i_j})}(c_j)$ to obtain Mek.

Also in the same paper [21], they define a security notion for broadcast encryption in the subset-cover framework namely, *Key Indistinguishability* (kIND) and prove that BE in the subset-cover framework is a secure broadcast encryption if it holds kIND and the corresponding encryption scheme E and F are IND-CCA1 secure. Therefore when proving the security of such a BE which is constructed in this framework, we just prove that it holds kIND. Informally, kIND says that any polynomial-time adversary can but with a negligible probability distinguish the subset key of privileged subset P of its choice from a random string of the same length even getting to know all private keys of users outside P .

Definition 3 (KEY PREDISTRIBUTION SCHEME, KPS). *A Key Predistribution Scheme KPS consists of a polynomial-time algorithms KeyGen, where:*

- KPS.Keygen($1^\lambda, n$) Takes as input a security parameter 1^λ , the number of users n . It outputs n sets of user keys I_1, \dots, I_n such that for $S \subseteq N$ and $S \neq \emptyset$, a conference key $\mathbf{k}(S)$ can be derived from I_u if and only if $u \in S$.

Observe that we can use KPS.Keygen for BE.Keygen resulting in a broadcast encryption in the subset-cover framework in which \mathcal{S} is a collection of all non-empty subsets of N . Consequently, we just let $\text{BE.Encrypt}(P, T, \text{Mek})$ output nothing (zero-header) and let Mek to be $\mathbf{k}(P)$. In this sense, KPS can be viewed as zero-header BE. Therefore, the security notion for KPS is indeed the key indistinguishability notion mentioned before.

3 Broadcast Encryption from PRSG

3.1 Generic Broadcast Encryption

We formally capture the nature of broadcast encryption scheme which is constructed from pseudo-random sequence generator into a general “pattern”. We call such a pattern **Sequential Key Derivation Pattern** or SKDP. This pattern was actually explained briefly in the introduction as the type 4 method to shorten the private key size. We formalize it here. We begin by giving the definition of this pattern first and explain later.

Definition 4 (SEQUENTIAL KEY DERIVATION PATTERN, SKDP). *Let $N := \{1, 2, \dots, n\}$. Let Γ be a forest of rooted trees in which each node is labelled a different subset of N . We say that (N, Γ) is a sequential key derivation pattern if:*

1. *The label at each node which is not a root in each tree is a superset of the label at its parent node.*
2. *For every subset S of N , S can be partitioned into a disjointed union of subsets labelled at some nodes in Γ .*

Notation. A forest Γ is specified by a set of nodes and a set of edges. Since each node is labelled a different subset of N , we represent a node as its label. One edge is defined by an ordered pair of nodes directed toward from their root. A path from root to leaf is defined by an ordered set of nodes on that path directed toward from their root. We will call a path from root to leaf a *rl-path*. The set of all rl-paths in Γ is denoted by $\text{Path}(\Gamma)$. An i -th node from root in rl-path \mathbf{a} is denoted by $\mathbf{a}[i]$. Observe that for $\mathbf{a} \in \text{Path}(\Gamma)$ it is true from the property 1 that $\mathbf{a}[i-1] \subset \mathbf{a}[i]$, thus we denote $\mathbf{a}[i] \setminus \mathbf{a}[i-1]$ by $\Delta\mathbf{a}[i]$ and call it a *differential label* at node $\mathbf{a}[i]$ for $i \geq 1$. Let $\Delta\mathbf{a}[0] = \mathbf{a}[0]$. For $\mathbf{a} \in \text{Path}(\Gamma)$ define a *differential path* $\Delta\mathbf{a}$ as an ordered set of all differential labels in the rl-path \mathbf{a} . Denote the set of all differential paths in Γ by $\text{DPath}(\Gamma)$. Denote ∇ the opposite operation of Δ , i.e., if $\mathbf{p} = \Delta\mathbf{a}$ then $\nabla\mathbf{p} = \mathbf{a}$ for $\mathbf{a} \in \text{Path}(\Gamma)$, $\mathbf{p} \in \text{DPath}(\Gamma)$. For clarity, note that $\nabla\mathbf{p}[i] = (\nabla\mathbf{p})[i]$. Denote $\mathbf{a}_{\vdash i}$ the ordered set in which elements are taken from the first i elements of \mathbf{a} in the same order. Also we often call a label an absolute label to distinguish it from a differential one.

An example of SKDP is the one shown as type 4 in Figure 1. A path $2 \rightarrow 12 \rightarrow 123$ is an example of rl-path. We represent it as $\mathbf{a} = (\{2\}, \{1, 2\}, \{1, 2, 3\})$. The corresponding differential path is thus $\Delta\mathbf{a} = (\{2\}, \{1\}, \{3\})$.

Intuitively, each node is assigned a subset key of its label (recall that its label is a subset of N). Informally, the property 1 in the above definition allows the one-way computation from a subset key of a node say v to subset keys of its child nodes say v_1, \dots, v_d . The property 2 makes sure that the subset-cover algorithm can be used. Note that in the following generic scheme, we will not use subset key to compute another subset key directly but will use an *intermediate key* as we will see later.

The cryptographic primitive that is used for one-way computation is *pseudo-random sequence generator* $G_d()$, $d \in \mathbb{Z}^+$ that d -ples the input, i.e., whose output length is d times the length of the input. We say that $G_d : \{0, 1\}^\lambda \mapsto \{0, 1\}^{d\lambda}$ is a pseudo-random sequence generator if no polynomial-time adversary can distinguish the output of G_t on a random chosen seed from a truly random chosen string of similar length.

Generic Construction. Now we will formally describe the generic broadcast encryption in the subset-cover framework that makes use of SKDP (N, Γ) . It is enough to specify only **SC.DefineSet**, **SC.Cover**, and **BE.Keygen** since **BE.Encrypt** and **BE.Decrypt** can be applied transparently from the last section.

SC.Defineset(n) It defines \mathcal{S} as the sets of labels at all nodes in Γ . It also output Γ as given from SKDP.

SC.Cover(P, \mathcal{S}) Due to the property 2 of SKDP, each subset P of N can be partitioned into a disjointed union of subsets labelled at some nodes in Γ , thus a disjointed union of subsets in \mathcal{S} . It partitions the set P into $\mathcal{S}_P := \{S_{i_1}, S_{i_2}, \dots, S_{i_m} : S_{i_j} \in \mathcal{S}\}$ with the minimum numbers of subsets.

BE.Keygen($1^\lambda, n$) Before specify the algorithm, the definitions of intermediate key, subset key, and their relation are specified first as follows:

INTERMEDIATE KEY. Each subset $S_i \in \mathcal{S}$ is assigned an *intermediate key* $\mathbf{t}(S_i)$. Each user in S_i , say u , should be able to derive $\mathbf{t}(S_i)$ from I_u .

SUBSET KEY. Each subset $S_i \in \mathcal{S}$ is assigned a *subset key* $\mathbf{k}(S_i)$. A subset key $\mathbf{k}(S_i)$ can be derived from the intermediate key $\mathbf{t}(S_i)$. We say that a node is assigned a subset key $\mathbf{k}(S_i)$ if that node is labelled S_i .

DERIVATION. Let S_i be a subset labelled at a node which is not a leaf in Γ . Suppose that the outdegree of this node is d . Let $S_{i_1}, S_{i_2}, \dots, S_{i_d}$ be subsets labelled at its children and $i_1 < \dots < i_d$. The derivation is defined as:

$$\mathbf{t}(S_{i_1}) || \mathbf{t}(S_{i_2}) || \dots || \mathbf{t}(S_{i_d}) || \mathbf{k}(S_i) := G_{d+1}(\mathbf{t}(S_i))$$

where $|\mathbf{t}(S_{i_1})| = \dots = |\mathbf{t}(S_{i_d})| = |\mathbf{k}(S_i)| = \lambda$ bits and $||$ is concatenation. This recurrence relation is well defined if all the initial values, which in fact are all the intermediate keys assigned at root of unconnected trees of the forest, are defined.

Now we will specify **BE.Keygen**. It randomly chooses λ -bits strings in exactly the same number as the number of unconnected trees in Γ . It then assigns each string to be the intermediate key assigned at the root of each unconnected tree respectively. User u should be given the intermediate key

assigned at the node whose label contains u , say node v , which appears first when looking from root to leaf in such a rl-path so that u can derive all the intermediate keys assigned at v 's descendants, whose labels are some supersets of label at v ; but not the ones assigned at v 's ancestors, whose labels do not contain u . That is, I_u is the set of all intermediate keys at nodes whose differential labels contain u . Formally $I_u = \{\mathbf{t}(\mathbf{a}[i]) : u \in \Delta\mathbf{a}[i], \mathbf{a} \in \text{Path}(\Gamma)\}$.

Theorem 1 *The above generic broadcast encryption scheme from SKDP satisfies the KIND property assuming secure pseudo-random sequence generator.*

REMARK 1: A NOTE ON PUBLIC KEY EXTENSION. Public-key extension of our generic broadcast encryption from SKDP can be constructed *directly* by utilizing Hierarchical Identity-Based Encryption [14] with the hierarchical tree obtained by connecting all roots of unconnected tree in Γ of SKDP to a new central root. This is indeed the same method as proposed by Dodis-Fazio [10], but we believe that our interpretation provides a better understanding.

3.2 Flexible SD/LSD Broadcast Encryption

We construct schemes called Flexible SD/LSD to demonstrate the power of our generic method. In general, the following mechanism is just one example of conversion from *any* PRSG-based broadcast encryption schemes into schemes which yielding less private key size while maintaining exactly the same header size. Moreover, with a further adaptation, we can reduce also the header size by trading off the computational cost. In particular, we apply this conversion to the SD scheme [21] and the LSD scheme [16] to get the Flexible SD and LSD schemes (FSD/FLSD) respectively. We call them flexible since their structures came from a flexible generalization of the idea by our general method.

We assume that the reader is familiar with the SD/LSD scheme. Observe that the SD/LSD schemes are indeed two such patterns of SKDP. The SKDP which implicitly used in SD is shown explicitly in Figure 2(left). The SKDP for LSD scheme is just an adaptation of SD scheme with the absence of some labels from SD scheme.

The conversion is very simple. Intuitively we just split the differential label of each node which is not a singleton subset of N in the original scheme into a union of differential labels which are singleton subsets and connect them in an appropriate order (step 1). After step 1, some of rl-paths will become sub-paths (from root) of some other rl-paths. Observe that such repetition parts represent the same sets of absolute labels. Therefore we can delete those sub-paths from the collection of all rl-paths since doing this does not affect the collection of absolute labels, and thus also the header size. These deletions reduce the private key size. In step 3, absolute labels which are not absolute labels in the original schemes are combined until there is no such case. We do this since these labels are not needed for subset-cover algorithm. They would only make rl-paths too long, consequently increase the computational cost. However, as we will see later,

step 3 can be skipped and we will obtain a scheme which beside the private key size, the header size is also reduced. Note that the procedures in the following description are somewhat made redundant for a better understanding.

The FSD and FLSD Scheme. Let X be SD or LSD scheme. Let (N, Γ_X) be a SKDP for X scheme. The conversion from X scheme to FX scheme is done as follows:

Step 1 For each $\mathbf{p} \in \text{DPath}(\Gamma_X)$, let $|\mathbf{p}| = p$ and $|\mathbf{p}[i]| = k_i$ and do the following:

1. For $i : 0 \leq i \leq p-1$, parse $\mathbf{p}[i]$ as $\{a_{i,1}, \dots, a_{i,k_i}\}$ where $a_{i,1} < \dots < a_{i,k_i}$.
2. Define $f(\mathbf{p}) = (\{a_{0,1}\}, \dots, \{a_{0,k_0}\}, \{a_{1,1}\}, \dots, \{a_{1,k_1}\}, \dots, \{a_{p,1}\}, \dots, \{a_{p,k_p}\})$.

After all, let $A = \{f(\mathbf{p}) : \mathbf{p} \in \text{DPath}(\Gamma_X)\}$.

Step 2 For any $\mathbf{v}, \mathbf{w} \in A$ such that $\mathbf{v} = \mathbf{w}_{-|\mathbf{v}|}$, we decrement A to be $A \setminus \{\mathbf{v}\}$. Repeat this until there is no such case.

Step 3 For each $\mathbf{q} \in A$, if there is j such that $\nabla \mathbf{q}[j] \notin S_X$ then renew \mathbf{q} to be $(\mathbf{q}[0], \dots, \mathbf{q}[j-1], \mathbf{q}[j] \cup \mathbf{q}[j+1], \mathbf{q}[j+2], \dots, \mathbf{q}[|\mathbf{q}|-1])$. Repeat this until there is no such case.

Step 4 Finally we let $\text{DPath}(\Gamma_{FX}) = A$.

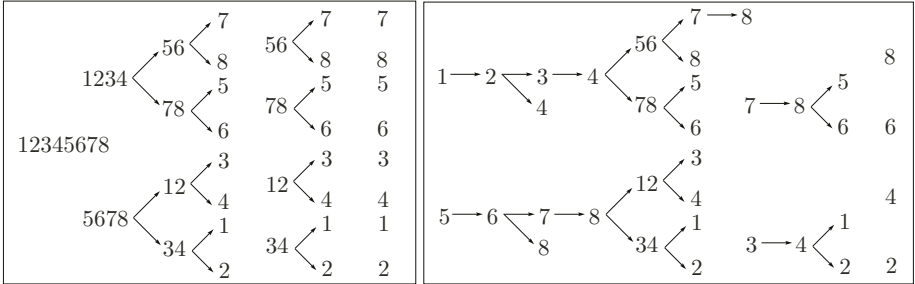


Fig. 2. Differential path representation of SKDP for SD (left) and FSD (right).

Theorem 2 For $1 \leq u \leq n-1$, let $x_u := \max\{k : 2^k \mid u\}$. Then

$$|I_{u,\text{FSD}}| = |I_{u,\text{SD}}| - \log n + x_u,$$

and $|I_{n,\text{FSD}}| = |I_{n,\text{SD}}|$. Recall that $|I_{u,\text{SD}}| = (\log^2 n + \log n)/2 + 1$ for all $u \in N$.

PROOF. It is enough to show how many differential paths containing u are deleted in step 2. We put the label N away for a while. For $1 \leq k \leq \log n$, let B_k be the collection of all k -length differential paths of SD. Wlog, we consider the deletions in $B_1, \dots, B_{\log n-1}$ respectively. By inspection, for each k , differential path $\mathbf{p} \in B_k$ will satisfy $f(\mathbf{p}) = f(\mathbf{p}')_{-|f(\mathbf{p})|}$ for some $\mathbf{p}' \in B_{k'}, k' > k$ iff

$$f(\mathbf{p}) = (\{q2^k + 1\}, \{q2^k + 2\}, \dots, \{q2^k + 2^k - 1\}),$$

for some $q \in \mathbb{Z}^+$. Thus there will be no deletion for user u iff $2^k \mid u$. Hence in the last step, when considering $k = \log n - 1$, the number of deletions for u will be $\log n - 1 - \max\{k : 2^k \mid u\} = \log n - 1 - x_u$. Finally, if $u \neq n$, then there is one more deletion from the label N thus the reduction will be $\log n - x_u$. \square

To illustrate this theorem, one can verify from Figure 2 that: in FSD, for $u = 1, 3, 5, 7 : |I_u| = 4$, for $u = 2, 6 : |I_u| = 5$, and $|I_4| = 6, |I_8| = 7$; while in SD, $|I_u| = 7$ for all $u \in N$. The following corollary follows directly from Theorem 2.

Corollary 1 *In the FSD scheme, for $1 \leq j \leq \log n$, there are exactly $n/2^j$ users whose the number of keys is $|I_{u,\text{SD}}| - \log n - 1 + j$. And only one remaining user has $|I_{u,\text{SD}}|$ keys.*

In the following theorem, we concern only the basic LSD (bLSD) scheme for simplicity. The results for general LSD schemes can be obtained similarly.

Theorem 3 *For $1 \leq u \leq n - 1$, let $x_u := \max\{k : 2^k \mid u\}$. Then*

$$|I_{u,\text{FbLSD}}| = |I_{u,\text{bLSD}}| - \log n + x_u + y_u,$$

where

$$y_u = |\{j : 1 \leq j \leq \log n, \sqrt{\log n} \nmid j, 2^j - 2^{\lfloor \frac{j}{\sqrt{\log n}} \rfloor \sqrt{\log n}} + 1 \leq u \bmod 2^j \leq 2^j - 1\}|,$$

and $|I_{n,\text{FbLSD}}| = |I_{n,\text{bLSD}}|$. Recall that $|I_{u,\text{bLSD}}| = \log^{3/2} n + 1$ for all $u \in N$.

Intuitively, the term y_u comes from the number of differential labels containing u that were in \mathcal{S}_{SD} and would have been deleted in step 2, but are not in $\mathcal{S}_{\text{bLSD}}$. Note that $y_u \leq \log n - x_u$ since we just add back what we would have deleted if it were SD scheme.

Analogous to Corollary 1, in the FbLSD scheme we could indeed show the exact number of users in the function of the number of keys. However, it turns out that the expressions are quite complex. We thus state only a particular case when the number of keys is the fewest to give some intuition as the following.

Corollary 2 *In the FbLSD scheme, there are exactly $n/2^{\sqrt{\log n}}$ users whose the number of keys is $|I_{u,\text{bLSD}}| - \log n$.*

Theorem 4 *The FSD/FLSD scheme require the same header size as the original scheme for every instance and the computational cost bounded by $O(\log^2 n)$.*

We briefly prove this theorem. First, the header size is remain unchanged since $\mathcal{S}_X = \mathcal{S}_{\text{FX}}$. Second, it can be shown that the longest rl-path contains $(\log^2 n + \log n)/2$ edges which is $O(\log^2 n)$.

REMARK 2: ON REDUCING THE HEADER SIZE. If we skip step 3, then there are some labels which are in \mathcal{S}_{FX} but not in \mathcal{S}_X . This means that in the FX scheme we have more choices to cover any privileged subsets hence the header size will be reduced for some instances of broadcast. On the down side, the longest rl-path will have length n resulting in increasing computational cost. Nevertheless, to skip or not to skip step 3 are the two extreme cases on the spectrum. In this sense, we can trade off the header size and the computational cost.

4 Key Predistribution Scheme from PRSG

4.1 Lower Bound

Theorem 5 *Every PRSG-based KPS satisfies $\max_{u \in N} |I_u| \geq \lceil (2^n - 1)/n \rceil$.*

Note that this is $O(1/n)$ less than the lower bound in the information-theoretically secure KPS in the same setting in which such a bound is 2^{n-1} [9].

A further fact from the proof of this theorem is that two necessary conditions to make an equality holds are: (i) The differential label at each node must be a singleton set. (ii) For each $u \in N$, $|I_u|$ is equal to $\lceil (2^n - 1)/n \rceil$ or $\lfloor (2^n - 1)/n \rfloor$.

4.2 An Optimal Construction

Intuition. To design a set of differential path $\text{DPath}(\Gamma)$ to represent a SKDP for an optimal KPS, we have to make sure the following requirements¹:

- It really represents a SKDP: the (absolute) labels at any two different nodes are different. For example, there is no such path from root (not necessary to leaf) $1 \rightarrow 12$ and $2 \rightarrow 12$ appear simultaneously; that is to say there is no differential path $1 \rightarrow 2 \rightarrow \dots$ and $2 \rightarrow 1 \rightarrow \dots$ at the same time.
- It can be used for KPS: the set of (absolute) labels at all nodes completes the set of non-empty subsets of N .
- It is optimal: for each $u \in N$ the number of differential labels which u appears is $\lceil (2^n - 1)/n \rceil$ or $\lfloor (2^n - 1)/n \rfloor$.

Consider the case where n is a prime larger than 2. When n is not a prime, a construction can be achieved similarly but is more complex. Theorem 5 can be intuitively interpreted as the following: First due to the fact (i), for each nonempty $S \subseteq N$ it must be that $\mathbf{t}(S) \in I_u$ for only one *unique* u . If we put $\mathbf{t}(N)$ away for a while. All other $2^n - 2$ intermediate keys for non empty-subset of N must be distributed equally to each u so that $|I_u| = \lfloor (2^n - 1)/n \rfloor = (2^n - 2)/n$. Note that this is an integer due to the Fermat's little theorem. Finally we pick one unlucky user say v and increment I_v to be $I_v \cup \{\mathbf{t}(N)\}$ so that only v , $|I_v| = \lceil (2^n - 1)/n \rceil = (2^n - 2)/n + 1$ and we will be done since this is optimal.

The question now is how to distribute $2^n - 2$ values of $\mathbf{t}(S), S \subset N, S \neq \emptyset$ equally to each $I_u, u \in N$. We accomplish this by first constructing a structure called *block*. One block contains n fix-length differential paths in which labels are all different. Each block has a property that the number of differential labels which u appears is equal for every $u \in N$. Thus we just let $\text{DPath}(\Gamma)$ to be composed of many blocks so that we would accomplish the task. However, the difficulty arises since we have to make sure also that any absolute labels from different blocks of the same length are different (as the requirement 1) and the set of all differential paths of length i completes the set of i -subsets of N (as requirement 2). This turns out to be a non-trivial task. We achieve this by

¹ Figure 4(lower part) should be helpful to get some insight.

defining an equivalence relation between blocks of the same length: informally two blocks are said to be equivalent if the set of labels from two blocks are the same. Therefore we just pick one block from each equivalence class into which this relation partitioned to completes $\text{DPath}(F)$ and we will be done. However, to pick a i -length block, it has to be consistent with some a $(i - 1)$ -length block picked previously in the sense that all absolute labels of the first i nodes away from the root in each path in two blocks are the same. To accomplish this, we define a relation called *splitting relation* between every two complete collections of classes in which length of blocks are consecutive, say $i - 1$ and i . This relation will imply a set of chains that relate from block of length 1 to that of length 2 and so on. Consequently, instead of picking up a block, we will pick up a chain and we will be done. Note that due to (i), wlog, from now on we consider each differential label as an element in N instead of a singleton subset of N .

Building Blocks. Now we will formally define the structure “block” and its equivalence relation. Each block is generated by a vector from the space $D_k := \{(d_1, \dots, d_k) \in (N_*)^k : \forall a < b, \sum_{j=a}^b d_j \not\equiv 0 \pmod{n}\}$ where we let $N_* = \{1, \dots, n - 1\}$. Figure 4(upper part) shows examples of block.

Definition 5 (BLOCK) For a set $F \subseteq N$ and a vector $\mathbf{d} = (d_1, \dots, d_{k-1}) \in D_{k-1}$, we define a block generated by \mathbf{d} over F as

$$\langle \mathbf{d} \rangle_F = \{(a, a + d_1, a + d_1 + d_2, \dots, a + d_1 + \dots + d_{k-1}) \bmod n : a \in F\},$$

and denote it as $\langle \mathbf{d} \rangle_F$. When $F = N$, we simply denote $\langle \mathbf{d} \rangle$.

Recall that each element in a block is a differential path whose all differential labels are singleton subset of N so we can treat such a differential path as a vector for simplicity. Furthermore one can verify that if n is prime, no absolute labels from any different differential paths in the same block are the same.

Definition 6 (EQUIVALENCE RELATION \equiv) For vectors $\mathbf{d}, \mathbf{e} \in D_{k-1}$, we say that \mathbf{d} is equivalent to \mathbf{e} over set F , and write $\mathbf{d} \equiv_F \mathbf{e}$, if

$$\{\nabla \mathbf{p} : \mathbf{p} \in \langle \mathbf{d} \rangle_F\} = \{\nabla \mathbf{p} : \mathbf{p} \in \langle \mathbf{e} \rangle_F\},$$

and when $F = N$, we simply denote $\mathbf{d} \equiv \mathbf{e}$.

It is easy to verified that \equiv is an equivalence relation on the set D_{k-1} , i.e., it has reflexivity, symmetry, and transitivity. So what will the equivalence classes into which the relation \equiv partitions D_{k-1} be like? First, we consider the following lemma. Let Π_k denote a set of all permutations of $(0, 1, \dots, k - 1)$.

Lemma 1 For $\mathbf{d} = (d_1, \dots, d_{k-1}), \mathbf{e} = (e_1, \dots, e_{k-1}) \in D_{k-1}$ let $d_0 := n - \sum_{j=1}^{k-1} d_j \bmod n, e_0 := n - \sum_{j=1}^{k-1} e_j \bmod n$, we have $\mathbf{d} \equiv \mathbf{e}$ if and only if there exists $(r_0, r_1, \dots, r_{k-1}) \in \Pi_k$ such that for all $0 \leq i \leq k - 1$, $e_i = \sum_{j=r_i}^{r_{i+1}-1 \bmod k} d_j \bmod n$, where we let $r_k := r_0$ and $\sum_{j=b}^a d_j := d_b + d_{b+1} + \dots + d_{k-1} + d_0 + \dots + d_a$ when $a < b$.

Definition 7 For $\mathbf{r} \in \Pi_k$, $\mathbf{d}' = (d_0, \dots, d_{k-1}) \in (N_*)^k$ such that $\sum_{j=0}^{k-1} d_j \equiv 0 \pmod{n}$, let

$$\mathbf{d}' \triangleright \mathbf{r} := \left(\sum_{j=r_0}^{r_1-1 \bmod k} d_j, \sum_{j=r_1}^{r_2-1 \bmod k} d_j, \dots, \sum_{j=r_{k-1}}^{r_0-1 \bmod k} d_j \right) \bmod n.$$

Lemma 1 implies that such an equivalence class is of the form $[\mathbf{d}'] := \{\mathbf{d}' \triangleright \mathbf{r} : \mathbf{r} \in \Pi_k\}$. To see the concrete classes, we first consider \mathbf{d}', \mathbf{e}' such that $\sum_{j=1}^{k-1} d_j = \sum_{j=1}^{k-1} e_j = n$. One can verify that $[\mathbf{d}'] = [\mathbf{e}']$ if and only if \mathbf{d}' is a cyclic permutation of \mathbf{e}' . Next observe that for an arbitrary $\mathbf{d} \in D_{k-1}$ there will be $\mathbf{r} \in \Pi_k$ and $\mathbf{v} = (v_0, \dots, v_{k-1}) \in (N_*)^k$ where $\sum_{j=1}^{k-1} v_j = n$ such that $\mathbf{d} \in [\mathbf{v} \triangleright \mathbf{r}]$. Therefore the complete collection of these equivalence classes is the collection of classes $[\mathbf{v}]$ where each \mathbf{v} is a *cyclic positive k -partition* of n . We denote this collection of equivalence classes as $\mathcal{E}_{n,k}$. For example, $\mathcal{E}_{6,3} = \{[411], [321], [312], [222]\}$.

From now, if we write $[\mathbf{v}] \in \mathcal{E}_{n,k}$ it is to be understood that $\mathbf{v} = (v_0, \dots, v_{k-1}) \in (N_*)^k$ and $\sum_{j=0}^{k-1} v_j \equiv n \pmod{n}$ unless something else specified; in addition we will say that \mathbf{v} is a representative vector of class $[\mathbf{v}]$.

Definition 8 (SPLITTING RELATION) A *splitting relation* $\mathbf{Spl}\mathbf{t}_k \subset \mathcal{E}_{n,k} \times \mathcal{E}_{n,k+1}$ is defined as $\mathbf{Spl}\mathbf{t}_k := \{([\mathbf{v}], [\mathbf{y}]) : [\mathbf{v} \dashv_{k-1} \|(v_{k-1} - a \bmod n, a)] = [\mathbf{y}], a \in \mathbb{Z}_{n-1}\}$.

This definition is well defined: we do not aware which representative vector of such a class is to be splitted, i.e., we claim the following lemma:

Lemma 2 For any \mathbf{w} such that $\mathbf{w} = \mathbf{v} \triangleright \mathbf{r}$ for some $\mathbf{r} \in \Pi_k$ there will be $\mathbf{s} \in \Pi_{k+1}$ and $b \in \{1, \dots, n-1\}$ such that $\mathbf{w} \dashv_{k-1} \|(w_{k-1} - b \bmod n, b) = (\mathbf{v} \dashv_{k-1} \|(v_{k-1} - a \bmod n, a)) \triangleright \mathbf{s}$.

To prove this Lemma, choose $\mathbf{s} = \mathbf{r} \|(k)$ and $b = w_{k-1} - v_{k-1} + a \bmod n$.

Lemma 3 For $2 \leq k \leq \lceil n/2 \rceil$ there exists onto function $f_{n,k} : \mathcal{E}_{n,k} \xrightarrow{\text{onto}} \mathcal{E}_{n,k-1}$ such that for all $[\mathbf{v}] \in \mathcal{E}_{n,k}$, $(f_{n,k}([\mathbf{v}]), [\mathbf{v}]) \in \mathbf{Spl}\mathbf{t}_{k-1}$. For $\lceil n/2 \rceil + 1 \leq k \leq n-1$ there exists one-to-one function $g_{n,k} : \mathcal{E}_{n,k} \xrightarrow{1-1} \mathcal{E}_{n,k-1}$ such that $g_{n,k}^{-1} \subset \mathbf{Spl}\mathbf{t}_{k-1}$.

For the functions denoted above, when $f_{n,k}([\mathbf{v}^k]) = [\mathbf{v}^{k-1}]$ ($2 \leq k \leq \lceil n/2 \rceil$) or $g_{n,k}([\mathbf{v}^k]) = [\mathbf{v}^{k-1}]$ ($\lceil n/2 \rceil + 1 \leq k \leq n-1$), we will represent it as $[\mathbf{v}^{k-1}] \rightarrow [\mathbf{v}^k]$. Let $C = \{f_{n,2}, \dots, f_{n,\lceil n/2 \rceil}, g_{n,\lceil n/2 \rceil+1}, \dots, g_{n,n-1}\}$. A chain $[\mathbf{v}^{k_1}] \rightarrow [\mathbf{v}^{k_1+1}] \rightarrow \dots \rightarrow [\mathbf{v}^{k_2}]$ is said to be induced by C if every \rightarrow in the chain is taken from a mapping in a function in C . Such a chain is said to be (k_1, k_2) -*terminated* if there is no \rightarrow directed into $[\mathbf{v}^{k_1}]$ and no \rightarrow directed from $[\mathbf{v}^{k_2}]$. Since $f_{n,k}$ is onto function, each terminated chain is $(1, k)$ -terminated for some $\lceil n/2 \rceil \leq k \leq n-1$.

The Optimal Construction

Step 1 Find a set of functions $A = \{f_{n,k} : \mathcal{E}_{n,k} \xrightarrow{\text{onto}} \mathcal{E}_{n,k-1} \mid 2 \leq k \leq \lceil n/2 \rceil\}$ and

$B = \{g_{n,k} : \mathcal{E}_{n,k} \xrightarrow{1-1} \mathcal{E}_{n,k-1} \mid \lceil n/2 \rceil + 1 \leq k \leq n-1\}$ which satisfy Lemma 3.

Step 2 For each terminated chain $[\mathbf{v}^1] \rightarrow [\mathbf{v}^2] \rightarrow \cdots \rightarrow [\mathbf{v}^k]$, we convert for $j : 1 \leq j \leq k$ each representation vector \mathbf{v}^j to \mathbf{w}^j so that $[\mathbf{v}^j] = [\mathbf{w}^j]$ and for $j : 1 \leq j \leq k-1$,

$$\mathbf{w}_{-j-1}^j \parallel (w_{j-1}^j - a_j \bmod n, a_j) = \mathbf{w}^{j+1},$$

for some $a_j \in \{1, \dots, n-1\}$. Note that we can do this due to Lemma 2 and the fact that there is only one \rightarrow directed into $[\mathbf{v}^{j+1}]$ because a function mapped from it determines the \rightarrow directed into it.

Step 3 Recall that $A \cup B$ induces only $(1, k)$ -terminated chains for some $\lceil n/2 \rceil \leq k \leq n-1$. Let \mathbf{ChnLst} be a set of all the last terms of terminated chains. Now we construct Γ by letting

$$\mathbf{DPath}(\Gamma) = \bigsqcup_{[\mathbf{x}] \in \mathbf{ChnLst}} \langle \mathbf{x}_{-1} \mid [\mathbf{x}]_{-1} \rangle.$$

Step 4 Pick one $(n-1)$ -length differential path in $\mathbf{DPath}(\Gamma)$, say \mathbf{p} . Increment it to $\mathbf{p} \parallel (a)$ where $a \notin \nabla \mathbf{p}$.

Theorem 6 For Γ above, we have that (N, Γ) is an SKDP with $\max_{u \in N} |I_u| = \lfloor (2^n - 1)/n \rfloor$ and the set of all labels of nodes completes the collection of all non-empty subsets of N .

An Example. $n = 7$. The diagram of chains induced by $A \cup B$ in step 1 is shown in Figure 3(left). After conversion in step 2 we have a diagram in Figure 3(right). $\mathbf{DPath}(\Gamma)$ defined in step 3 is $\langle (1, 1, 1, 1, 1) \rangle \sqcup \langle (3, 5, 5, 6) \rangle \sqcup \langle (2, 1, 1, 2) \rangle \sqcup \langle (3, 1, 2) \rangle \sqcup \langle (2, 2, 1) \rangle$. Lastly we pick $(1, 2, 3, 4, 5, 6) \in \mathbf{DPath}(\Gamma)$ and increment it to be $(1, 2, 3, 4, 5, 6, 7)$. This yields $|I_u| = \lfloor (2^7 - 1)/7 \rfloor = 18$ for $u \neq 7$ and $|I_7| = 19$.

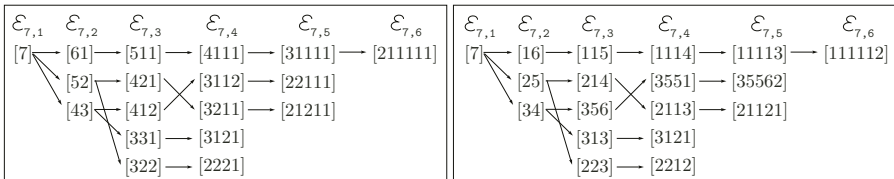


Fig. 3. Diagram of chains induced by $A \cup B$ in step 1(left) and its conversion after step 2(right). Recall that the direction of \rightarrow is opposite to the way functions are mapped.

$$\mathbf{p}_{1,1} = (1, 2, 3, 4, 5, 6, 7)$$

$$\left\{ \begin{array}{l} \mathbf{p}_{1,2} = (2, 3, 4, 5, 6, 7) \\ \mathbf{p}_{1,3} = (3, 4, 5, 6, 7, 1) \\ \mathbf{p}_{1,4} = (4, 5, 6, 7, 1, 2) \\ \mathbf{p}_{1,5} = (5, 6, 7, 1, 2, 3) \\ \mathbf{p}_{1,6} = (6, 7, 1, 2, 3, 4) \\ \mathbf{p}_{1,7} = (7, 1, 2, 3, 4, 5) \end{array} \right\} = \langle (1, 1, 1, 1, 1) \rangle \setminus \{(1, 2, 3, 4, 5, 6)\}$$

$$\left\{ \begin{array}{l} \mathbf{p}_{2,1} = (1, 4, 2, 7, 6) \\ \mathbf{p}_{2,2} = (2, 5, 3, 1, 7) \\ \mathbf{p}_{2,3} = (3, 6, 4, 2, 1) \\ \mathbf{p}_{2,4} = (4, 7, 5, 3, 2) \\ \mathbf{p}_{2,5} = (5, 1, 6, 4, 3) \\ \mathbf{p}_{2,6} = (6, 2, 7, 5, 4) \\ \mathbf{p}_{2,7} = (7, 3, 1, 6, 5) \end{array} \right\} = \langle (3, 5, 5, 6) \rangle$$

$$\left\{ \begin{array}{l} \mathbf{p}_{3,1} = (1, 3, 4, 5, 7) \\ \mathbf{p}_{3,2} = (2, 4, 5, 6, 1) \\ \mathbf{p}_{3,3} = (3, 5, 6, 7, 2) \\ \mathbf{p}_{3,4} = (4, 6, 7, 1, 3) \\ \mathbf{p}_{3,5} = (5, 7, 1, 2, 4) \\ \mathbf{p}_{3,6} = (6, 1, 2, 3, 5) \\ \mathbf{p}_{3,7} = (7, 2, 3, 4, 6) \end{array} \right\} = \langle (2, 1, 1, 2) \rangle$$

$$\left\{ \begin{array}{l} \mathbf{p}_{4,1} = (1, 4, 5, 7) \\ \mathbf{p}_{4,2} = (2, 5, 6, 1) \\ \mathbf{p}_{4,3} = (3, 6, 7, 2) \\ \mathbf{p}_{4,4} = (4, 7, 1, 3) \\ \mathbf{p}_{4,5} = (5, 1, 2, 4) \\ \mathbf{p}_{4,6} = (6, 2, 3, 5) \\ \mathbf{p}_{4,7} = (7, 3, 4, 6) \end{array} \right\} = \langle (3, 1, 2) \rangle$$

$$\left\{ \begin{array}{l} \mathbf{p}_{5,1} = (1, 3, 5, 6) \\ \mathbf{p}_{5,2} = (2, 4, 6, 7) \\ \mathbf{p}_{5,3} = (3, 5, 7, 1) \\ \mathbf{p}_{5,4} = (4, 6, 1, 2) \\ \mathbf{p}_{5,5} = (5, 7, 2, 3) \\ \mathbf{p}_{5,6} = (6, 1, 3, 4) \\ \mathbf{p}_{5,7} = (7, 2, 4, 5) \end{array} \right\} = \langle (2, 2, 1) \rangle$$

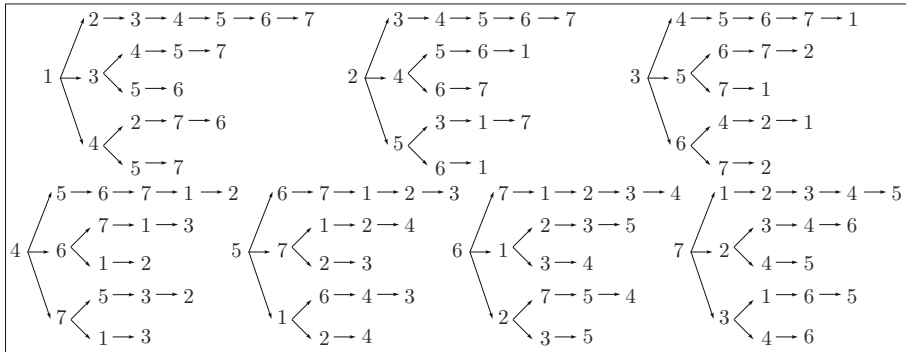


Fig. 4. $\text{DPPath}(\Gamma)$ (upper part) and its differential path representation (lower part) of SKDP for optimal KPS.

References

1. S. G. Akl and P. D. Taylor, “Cryptographic Solution to a Problem of Access Control in a Hierarchy”, ACM Transactions on Computer Systems, Vol. 1, No. 3, pp. 239-248, 1983.
2. J. Anzai, N. Matsuzaki and T. Matsumoto, “Quick Group Key Distribution Scheme with Entity Revocation”, Asiacrypt 1999, LNCS 1716, pp. 333-347, 1999.
3. T. Asano, “A Revocation Scheme with Minimal Storage at Receivers”, ASIACRYPT 2002, LNCS 2501, pp.433-450.

4. N. Attrapadung, K. Kobara, H. Imai, "Broadcast Encryption with Short Keys and Transmissions", ACM Workshop on Digital Rights Management, October 2003.
5. R. Blom, "An Optimal Class of Symmetric Key Generation Systems", EURO-CRYPT 1984, LNCS 209, pp. 335-338.
6. C. Blundo, L. F. Mattos, D. R. Stinson. "Trade-offs Between communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution", CRYPTO'96, LNCS 1109 pp. 387-400.
7. D. Boneh and A. Silverberg, "Applications of Multilinear Forms to Cryptography", preprint, 2002. Available from <http://eprint.iacr.org>.
8. G. C. Chick and S. E. Tavares, "Flexible Access Control with Master Keys", Crypto 1989, LNCS 435, pp. 316-322, 1990.
9. Y. Desmedt, V. Viswanathan, "Unconditionally Secure Dynamic Conference Key Distribution", IEEE, ISIT'98.
10. Y. Dodis and N. Fazio, "Public Key Broadcast Encryption for Stateless Receivers", ACM Workshop on Digital Rights Management, November 2002.
11. Y. Dodis and N. Fazio, "Public Key Broadcast Encryption Secure against Adaptive Chosen Ciphertext Attack", PKC '03, LNCS 2567, 2003.
12. Y. Dodis, N. Fazio, A. Kiayias, M. Yung, "Fully Scalable Public-Key Traitor Tracing", To appear in the proceeding of PODC 2003.
13. A. Fiat, M. Naor, "Broadcast Encryption", CRYPTO 1993, LNCS 0773, pp. 480-491.
14. C. Gentry, A. Silverberg, "Hierarchical ID-Based Cryptography", ASIACRYPT 2002, LNCS 2501, pp.548-566.
15. E. Gafni, J. Staddon, Y.L.Yin, "Efficient Methods for Integrating Traceability and Broadcast Encryption", CRYPTO 1999, LNCS 1666, pp. 372-387.
16. D. Halevi, A. Shamir "The LSD Broadcast Encryption Scheme", CRYPTO 2002, LNCS 2442, pp. 47-60.
17. R. Kumar, S. Rajagopalan, A. Sahai, "Coding Constructions for Blacklisting Problems without Computational Assumptions", CRYPTO 1999, LNCS 1666, pp. 609-623.
18. K. Kurosawa, T. Yoshida, Y. Desmedt, M. Burmester, "Some Bounds and a Construction for Secure Broadcast Encryption", ASIACRYPT'98, LNCS 1514, pp. 420-433.
19. M. Luby, J. Staddon, "Combinatorial Bounds for Broadcast Encryption", EURO-CRYPT 1998, LNCS 1403, pp. 512-526.
20. T. Matsumoto, H. Imai, "On the Key Predistribution System: A Practical Solution to the Key Distribution Problem", CRYPTO'87, 185-193.
21. D. Naor, M. Naor and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", CRYPTO 2001, LNCS 2139, 41-62.
22. M. Naor and B. Pinkas, "Efficient Trace and Revoke Schemes", Financial Cryptography FC 2000, LNCS1962, pp.1-20.
23. D.R.Stinson, "On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption", Designs, Codes and Cryptography 12 (1997), 215-243.
24. D. Wallner, E. Harder and R. Agee, "Key Management for Multicast: Issues and Architectures", IETF NetworkWorking Group, Request for Comments: 2627, available from <ftp://ftp.ietf.org/rfc/rfc2627.txt>, 1999.
25. C. K. Wong, M. Gouda and S. S. Lam, "Secure Group Communications Using Key Graphs", ACM SIGCOMM'98, 1998.

Boneh *et al.*'s k -Element Aggregate Extraction Assumption Is Equivalent to the Diffie-Hellman Assumption

Jean-Sebastien Coron and David Naccache

Gemplus Card International

34, rue Guynemer, Issy-les-Moulineaux, F-92447, France
{jean-sebastien.coron,david.naccache}@gemplus.com

Abstract. In Eurocrypt 2003, Boneh *et al.* presented a novel cryptographic primitive called *aggregate signatures*. An aggregate signature scheme is a digital signature that supports aggregation: i.e. given k signatures on k distinct messages from k different users it is possible to aggregate all these signatures into a single short signature. Applying the above concept to verifiably encrypted signatures, Boneh *et al.* introduced a new complexity assumption called *the k -Element Aggregate Extraction Problem*.

In this paper we show that the k -Element Aggregate Extraction Problem is nothing but a Computational Diffie-Hellman Problem in disguise.

Keywords: aggregate signatures, Diffie-Hellman problem, complexity assumption.

1 Introduction

In Eurocrypt 2003, Boneh, Gentry, Lynn and Shacham [2] introduced the concept of *aggregate signatures*. An aggregate signature scheme is a digital signature that supports aggregation: given k signatures on k distinct messages from k different users it is possible to aggregate all these signatures into a single short signature. This useful primitive allows to drastically reduce the size of public-key certificates, thereby saving storage and transmission bandwidth.

Applying the previous construction to verifiably encrypted signatures, Boneh *et al.* introduced in [2] a new complexity assumption called *the k -Element Aggregate Extraction Problem* (hereafter k -EAEP). In this paper we will prove that k -EAEP is equivalent to the Computational Diffie Hellman assumption (CDH).

This paper is structured as follows: section 2 recalls Boneh *et al.*'s setting, section 3 contains [2,3]'s definition of the k -EAEP and section 4 concludes the paper by proving the equivalence between k -EAEP and CDH.

2 Verifiable Encrypted Signatures via Aggregation

We will adopt [2,3]'s notations and settings, namely:

- G_1 and G_2 are two multiplicative cyclic groups of prime order p ;

- g_1 is a generator of G_1 and g_2 is a generator of G_2 ;
- ψ is a computable isomorphism from G_1 to G_2 with $\psi(g_1) = g_2$;
- e is a computable bilinear map $e : G_1 \times G_2 \rightarrow G_T$ where G_T is multiplicative and of order p . The map e is:
 - Bilinear: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$
 - Non-degenerate: $e(g_1, g_2) \neq 1$
- $h : \{0, 1\}^* \rightarrow G_2$ is a hash function.

Key generation	
	Pick random $x \xleftarrow{R} \mathbb{Z}_p$
	Compute $v \leftarrow g_1^x$
Public :	$v \in G_1$
Private :	$x \in \mathbb{Z}_p$
Signature	
	Hash the message $M \in \{0, 1\}^*$ into $h \leftarrow h(M) \in G_2$
	Compute the signature $\sigma \leftarrow h^x \in G_2$
Verification of σ (with respect to v and M)	
	Compute $h \leftarrow h(M)$
	Check that $e(g_1, \sigma) = e(v, h)$

Fig. 1. Boneh, Lynn, Shacham Signatures.

2.1 Boneh-Lynn-Shacham Signatures

Figure 1 briefly recalls Boneh, Lynn and Shacham's signature scheme [1], upon which the aggregate signatures schemes of [2,3] are based.

2.2 Aggregate Signatures

Consider now a set of k users using Figure 1's scheme (each user having a different key pair bearing an index i) and signing different messages M_i . Aggregation consists in combining the resulting k signatures $\{\sigma_1, \dots, \sigma_k\}$ into one aggregate signature σ . This is done by simply computing:

$$\sigma \leftarrow \prod_{i=1}^k \sigma_i$$

Aggregate verification is very simple and consists in checking that the M_i are mutually distinct and ensuring that:

$$e(g_1, \sigma) = \prod_{i=1}^k e(v_i, h_i) \quad \text{where } h_i = h(M_i)$$

This holds because:

$$e(g_1, \sigma) = e(g_1, \prod_{i=1}^k h_i^{x_i}) = \prod_{i=1}^k e(g_1, h_i)^{x_i} = \prod_{i=1}^k e(g_1^{x_i}, h_i) = \prod_{i=1}^k e(v_i, h_i)$$

2.3 Verifiably Encrypted Signatures via Aggregation

As explained in [2,3], verifiably encrypted signatures are used in contexts where Alice wants to show Bob that she has signed a message but does not want Bob to possess her signature on that message. Alice can achieve this by encrypting her signature using the public key of a trusted third party (*adjudicator*, hereafter Carol), and send the resulting ciphertext to Bob along with a proof that she has given him a valid encryption of her signature. Bob can verify that Alice has signed the message but cannot deduce any information about her signature. Later in the protocol, Bob can either obtain the signature from Alice or resort to the good offices of Carol who can reveal Alice's signature.

To turn the aggregate signature scheme into a verifiably encrypted signature scheme, [2,3] proceed as follows:

- Alice wishes to create a verifiably encrypted signature that Bob will verify, Carol being the adjudicator. Alice and Carol's keys are generated as if they were standard signers participating in the aggregate signature protocol described in the previous subsection.
- Alice creates a signature σ on M under her public key. She then forges a signature σ' on some random message M' under Carol's public key (we refer the reader to [2,3] for more details on the manner in which this existential forgery is produced). She then combines σ and σ' obtaining the aggregate ω . The verifiably encrypted signature is $\{\omega, M'\}$.
- Bob validates Alice's verifiably encrypted signature $\{\omega, M'\}$ on M by checking that ω is a valid aggregate signature by Alice on M and by Carol on M' .
- Carol adjudicates, given a verifiably encrypted signature $\{\omega, M'\}$ on M by Alice, by computing the signature σ' on M' and removing σ' from the aggregate thereby revealing Alice's signature σ .

3 The k -Element Aggregate Extraction Problem

As is clear, the security of Boneh *et al.*'s verifiable encrypted signature scheme depends on the assumption that given an aggregate signature of k signatures (here $k = 2$) it is difficult to extract from it the individual signatures (namely: Alice's signature on M). This is formally proved in theorem 3 of [2,3].

Considering the bilinear aggregate signature scheme on G_1 and G_2 , Boneh *et al.* assume that it is difficult to recover the individual signatures σ_i given the aggregate σ , the public-keys and the message digests. Actually, [2,3] assume that it is difficult to recover any aggregate σ' of any proper set of the signatures and term this the k -Element Aggregate Extraction Problem (hereafter k -EAEP).

More formally, this assumption is defined in [2,3] as follows: Let G_1 and G_2 be two multiplicative cyclic groups of prime order p , with respective generators g_1 and g_2 , a computable isomorphism $\psi : G_1 \rightarrow G_2$ such that $g_2 = \psi(g_1)$, and a computable bilinear map $e : G_1 \times G_2 \rightarrow G_T$.

Consider a k -user aggregate in this setting. Each user has a private key $x_i \in \mathbb{Z}_p$ and a public key $v_i = g_1^{x_i} \in G_1$. Each user selects a distinct message $M_i \in \{0, 1\}^*$ whose digest is $h_i \in G_2$ and creates a signature $\sigma_i = h_i^{x_i} \in G_2$. Finally, the signatures are aggregated yielding:

$$\sigma = \prod_{i=1}^k \sigma_i \in G_2$$

Let I be the set $\{1, \dots, k\}$. Each public-key v_i can be expressed as $g_1^{x_i}$, each digest h_i as $g_2^{y_i}$, each signature σ_i as $g_2^{x_i y_i}$ and the aggregate signature σ as g_2^z where:

$$z = \sum_{i \in I} x_i y_i$$

Definition 1 (k -EAEP). *The k -Element Aggregate Extraction Problem is the following: given the group elements $g_1^{x_1}, \dots, g_1^{x_k}, g_2^{y_1}, \dots, g_2^{y_k}$ and $g_2^{\sum_{i \in I} x_i y_i}$, output (σ', I') such that $I' \subsetneq I$ and $\sigma' = g_2^{\sum_{i \in I'} x_i y_i}$.*

The advantage of an algorithm \mathcal{E} in solving the k -EAEP is defined as:

$$\text{Adv } k\text{-Extr}_{\mathcal{E}} \stackrel{\text{def}}{=} \Pr \left[\begin{array}{l} (I' \subsetneq I) \wedge (\sigma' = g_2^{\sum_{i \in I'} x_i y_i}) : \\ x_1, \dots, x_k, y_1, \dots, y_k \xleftarrow{R} \mathbb{Z}_p, \sigma \leftarrow g_2^{\sum_{i \in I} x_i y_i}, \\ (\sigma', I') \xleftarrow{R} \mathcal{E}(g_1^{x_1}, \dots, g_1^{x_k}, g_2^{y_1}, \dots, g_2^{y_k}, \sigma) \end{array} \right]$$

wherein the probability is taken over the choices of all x_i and y_i and the coin tosses of \mathcal{E} .

In the following, we define the hardness of the k -EAEP. For simplicity, we use the asymptotic setting instead of the concrete setting of [2].

Definition 2. *The k -Element Aggregate Extraction Problem is said to be hard if no probabilistic polynomial-time algorithm can solve it with non-negligible advantage.*

[2,3] is particularly concerned with the case $k = 2$ where the aggregate extraction problem boils down to the following:

Definition 3 (2-EAEP). *Given $g_1^a, g_1^b, g_2^u, g_2^v$ and g_2^{au+bv} , output g_2^{au} .*

We refer the reader to [3] for more details on the manner in which this assumption is used in proving the security of the verifiable encrypted signature scheme.

4 k -EAEP Is Equivalent to the Computational Co-Diffie-Hellman Problem

The Computational co-Diffie-Hellman problem (hereafter co-CDH) is a natural generalization to two groups G_1 and G_2 of the standard Computational Diffie-Hellman problem; it is defined as follows [2]:

Definition 4 (co-CDH). Given $g_1, g_1^a \in G_1$ and $h \in G_2$, output $h^a \in G_2$.

The advantage of an algorithm \mathcal{A} in solving co-CDH in groups G_1 and G_2 is:

$$\text{Adv co-CDH}_{\mathcal{A}} \stackrel{\text{def}}{=} \Pr \left[\mathcal{A}(g_1, g_1^a, h) = h^a : a \xleftarrow{R} \mathbb{Z}_p, h \xleftarrow{R} G_2 \right]$$

The probability is taken over the choice of a , h and \mathcal{A} 's coin tosses. Note that when $G_1 = G_2$, this problem reduces to the standard CDH problem.

Definition 5. The Computational co-Diffie-Hellman problem in groups G_1 and G_2 is said to be hard if no probabilistic polynomial-time algorithm can solve it with non-negligible advantage.

The following theorem shows that the k -Element Aggregate Extraction Problem is equivalent to the Computational co-Diffie-Hellman problem.

Theorem 1. The k -Element Aggregate Extraction Problem is hard if and only if the Computational co-Diffie-Hellman problem is hard.

Proof. It is straightforward to show that an algorithm \mathcal{A} solving co-CDH can be used to solve the k -EAEP. Namely, given the instance $g_1^{x_1}, \dots, g_1^{x_k}, g_2^{y_1}, \dots, g_2^{y_k}$ and $g_2^{\sum_{i \in I} x_i \cdot y_i}$, using \mathcal{A} we obtain $\sigma' = g_2^{x_1 y_1}$ from $g_1, g_1^{x_1}, g_2^{y_1}$. This gives $(\{1\}, \sigma')$ as a solution to the k -EAEP.

For the converse, we start with $k = 2$, i.e. an algorithm solving the 2-EAEP and show how to generalize the method to arbitrary k . Letting g_1, g_1^a, g_2^u be a given instance of co-CDH, we must compute $g_2^{a \cdot u}$ using an algorithm \mathcal{A} solving the 2-EAEP.

We generate $x \xleftarrow{R} \mathbb{Z}_p$ and $y \xleftarrow{R} \mathbb{Z}_p$; one can see that:

$$(g_1^a, g_1^{a+x}, g_2^{-u}, g_2^{u+y}, g_2^{a \cdot y + u \cdot x + x \cdot y})$$

is a valid random instance of the 2-EAEP. The instance is valid because:

$$-a \cdot u + (a + x) \cdot (u + y) = a \cdot y + u \cdot x + x \cdot y$$

The instance is a random one because g_1^{a+x} and g_2^{u+y} are uniformly distributed in G_1 and G_2 . Moreover, the instance can be computed directly from g_2^u and $g_2^a = \psi(g_1^a)$. Therefore, given as input this instance, the algorithm \mathcal{A} outputs $g_2^{-a \cdot u}$, from which we compute $g_2^{a \cdot u}$ and solve the co-CDH problem.

More generally, for $k > 2$, we generate $x_2, \dots, x_k, y_2, \dots, y_k \xleftarrow{R} \mathbb{Z}_p$; then we generate the following instance of the k -EAEP:

$$(g_1^a, g_1^{a+x_2}, \dots, g_1^{a+x_k}, g_2^{-(k-1)u}, g_2^{u+y_2}, \dots, g_2^{u+y_k}, g_2^z)$$

where

$$z = \sum_{i=2}^k a \cdot y_i + x_i \cdot (u + y_i)$$

As previously, this is a valid random instance of the k -EAEP, which can be computed from g_2^u and $g_2^a = \psi(g_1^a)$. Therefore, given this instance as input, an algorithm \mathcal{A} solving k -EAEP outputs (I', σ') . We assume that $1 \in I'$, otherwise we can take $I'' \leftarrow I \setminus I'$ and $\sigma'' \leftarrow g_2^z / \sigma'$. Letting $\sigma' = g_2^{z'}$ and $k' = |I'| < k$, we have:

$$\begin{aligned} z' &= -(k-1) \cdot a \cdot u + \sum_{i \in I', i > 1} (a + x_i)(u + y_i) \\ z' &= a \cdot u \cdot (k' - k) + \sum_{i \in I', i > 1} a \cdot y_i + x_i \cdot (u + y_i) \end{aligned}$$

Therefore we can compute:

$$g_2^{a \cdot u} = \left(\sigma' \cdot \prod_{i \in I', i > 1} (g_2^a)^{-y_i} (g_2^u)^{-x_i} g_2^{-x_i y_i} \right)^{\frac{1}{k' - k}}$$

which is the solution of the co-CDH instance.

Therefore, given a polynomial time probabilistic algorithm solving the k -EAEP with non-negligible advantage, we obtain a polynomial time probabilistic algorithm solving co-CDH with non-negligible advantage, and conversely, with a tight reduction in both directions. \square

5 Conclusion

In this paper we showed that the k -element Aggregate Extraction Problem introduced by Boneh, Gentry, Lynn and Shacham in [2,3] is equivalent to the Computational Diffie Hellman Problem.

By shedding light on the connection between Boneh *et al.*'s verifiable encrypted signature scheme and the well-researched Computational Diffie-Hellman Problem, we show that [2,3] features, not only attractive computational requirements and short signature size, but also strong security assurances.

References

1. D. Boneh, B. Lynn and H. Shacham, *Short Signatures From the Weil Pairing*, Proceedings of ASIACRYPT' 2001, Lecture Notes in Computer Science vol. 2248, Springer-Verlag, pp. 514-532, 2001.
2. D. Boneh, C. Gentry, B. Lynn and H. Shacham, *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*, Advances in Cryptology - EUROCRYPT' 2003 Proceedings, Lecture Notes in Computer Science vol. 2656, E. Biham ed., Springer-Verlag, pp. 416-432, 2003.
3. D. Boneh, C. Gentry, B. Lynn and H. Shacham, *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*, Cryptology ePrint Archive, Report 2002/175, 2002, <http://eprint.iacr.org/>.

On Diophantine Complexity and Statistical Zero-Knowledge Arguments

Helger Lipmaa

Laboratory for Theoretical CS

Department of CS&E, Helsinki University of Technology

P.O.Box 5400, FIN-02015 HUT, Espoo, Finland

helger@tcs.hut.fi

Abstract. We show how to construct practical honest-verifier statistical zero-knowledge *Diophantine* arguments of knowledge (HVSZK AoK) that a committed tuple of integers belongs to an arbitrary language in bounded arithmetic. While doing this, we propose a new algorithm for computing the Lagrange representation of nonnegative integers and a new efficient representing polynomial for the exponential relation. We apply our results by constructing the most efficient known HVSZK AoK for non-negativity and the first constant-round practical HVSZK AoK for exponential relation. Finally, we propose the outsourcing model for cryptographic protocols and design communication-efficient versions of the Damgård-Jurik multi-candidate voting scheme and of the Lipmaa-Asokan-Niemi $(b + 1)$ st-price auction scheme that work in this model.

Keywords: Arguments of knowledge, Diophantine complexity, integer commitment scheme, statistical zero knowledge.

1 Introduction

A set $S \subset \mathbb{Z}^n$ is called *Diophantine* [Mat93], if it has a *representing polynomial* $\mathfrak{R}_S \in \mathbb{Z}[X; Y]$, $X = (X_1, \dots, X_n)$ and $Y = (Y_1, \dots, Y_m)$, such that $\mu \in S$ iff for some witness $\omega \in \mathbb{Z}^m$, $\mathfrak{R}_S(\mu; \omega) = 0$. A seminal result of Matiyasevich from 1970 states that every recursively enumerable set is Diophantine. It has been an open question since [AM76], whether $\mathbf{D} \stackrel{?}{=} \mathbf{NP}$, where \mathbf{D} is the class of sets S that have representing polynomials \mathfrak{R}_S , such that $\mu \in S$ iff for some polynomially long witness $\omega \in \mathbb{Z}^m$, $\mathfrak{R}_S(\mu; \omega) = 0$. One is also tempted to ask a similar question $\mathbf{PD} \stackrel{?}{=} \mathbf{P}$ about the “deterministic” version of class \mathbf{D} , the class \mathbf{PD} that contains such languages for which the corresponding polynomially-long witnesses can be found in polynomial time. The gap in our knowledge in such questions is quite surprising; this is maybe best demonstrated by the recent proof of Pollett that if $\mathbf{D} \subseteq \mathbf{co-NLOGTIME}$ then $\mathbf{D} = \mathbf{NP}$ [Pol03].

In this paper we take a more practice oriented approach. Namely, we are interested in the sets S with sub-quadratic (i.e., with length, sub-quadratic in the length of the inputs) witnesses. We propose representing polynomials with sub-quadratic, polynomial-time computable, witnesses for a practically important,

although relatively small, class L_2 of languages of bounded arithmetic. (This class of languages includes many arithmetic and number-theoretic relations like $[\mu_3 = \max(\mu_1, \mu_2)]$, but also relations like $[\mu_2 \text{ is the } i\text{th bit of } \mu_1]$.) For this, we demonstrate that the exponential relation has a representing polynomial with polynomial-time computable sub-quadratic-length witnesses. This improves somewhat on the previous best result of [AM76]; differently from the latter, we will also give a self-contained proof of this result, and provide a precise complexity analysis. Our next contribution is a new algorithm for finding, given a positive integer μ , such integers $(\omega_1, \dots, \omega_4)$ that $\mu = \omega_1^2 + \dots + \omega_4^2$. This algorithm improves on the Rabin-Shallit algorithm [RS86].

While representing polynomials with short witnesses have independent interest in complexity theory [AM76], our work on this topic was motivated by cryptographic applications. Given an integer commitment scheme [FO99,DF02] with efficient arguments of knowledge for additive and multiplicative relations, one can argue (by using the methodology from [FO99]) in honest-verifier statistical zero-knowledge (HVSZK) that $f(\mu) = 0$, where μ is a tuple of committed integers. By following this methodology, one can design efficient argument systems for several important cryptographic problems. However, there has been no previous formal treatment of what happens if one extends this methodology (at least not when coupled with an *integer* commitment scheme) so as to enable the demonstration of knowledge of an auxiliary witness ω , for which $f(\mu; \omega) = 0$. A natural requirement here is that if the arguer convinces the verifier that she knows such an ω , the verifier will also be convinced that $\mu \in S$ where $f = \mathfrak{R}_S$ is the representing polynomial of S .

Thus, by using well-known cryptographic tools, one can construct polynomial-length three-round HVSZK arguments of knowledge that $\mu \in S$ for any $S \in \mathbf{D}$. However, these arguments can only be executed if the arguer knows the corresponding witness. If there is a polynomial-time algorithm to compute the witness from μ (that is, $S \in \mathbf{PD}$), then one will be able to argue that $\mu \in S$ for an arbitrary $\mu \in S$. If, additionally, the corresponding witnesses are sub-quadratic (as they are when $S \in L_2$) then by using the described methodology one can often improve upon previously known arguments of knowledge—either in efficiency, or by basing the arguments on weaker security requirements: namely, it is sufficient to require that the underlying integer commitment scheme is statistically hiding and computationally binding [FO99]. In particular, we use our new algorithm for finding the representation $\mu = \omega_1^2 + \dots + \omega_4^2$ to propose a new argument of knowledge for non-negativity of the committed integer. Compared to Boudot's protocol for the same problem [Bou00], this argument is conceptually much simpler, somewhat shorter, and offers perfect completeness.

After that, we propose a general model for cryptographic protocols that involve social or financial choices (e.g., voting or auctions). In this model one can implement any function from the class L_2 (e.g., maximum-finding in the case of auctions) by using sub-quadratic-length interaction. As [CGS97,DJ01,LAN02], our model uses a certain encoding function enc of the social choices together with a homomorphic public-key cryptosystem. As an example, in this model we

can construct an efficient minimal-disclosure voting protocol where the talliers will only get to know the winning candidate.

Finally, we propose a few alternative constructions for the encoding function. Until now, one has mostly used the function $\text{enc}(n) = a^n$, where a is an a priori fixed upper limit on the number of participants [CGS97,DJ01,LAN02]. We show that instead, one can use the function $\text{enc}(n) = Z_a(n)$, where $Z_a(n)$ is the n th member of a certain Lucas sequence, to achieve otherwise exactly the same properties as in [DJ01,LAN02] but with correctness arguments of length $\Theta(\max(k, m \log a))$, where k is the security parameter, a is the maximal number of participants, and m is the number of possible social or financial choices (e.g., the number of different bids). This is $\Theta(\log m)$ times more efficient than the protocols from [DJ01,LAN02]. We also propose an efficient algorithm for computing $Z_a(n)$. Lucas sequences have definitely more applications in zero-knowledge proofs or arguments than described in this paper. We also demonstrate another approach that uses exponentiation as the encoding function.

Road-Map. We introduce necessary preliminaries in Section 2. In Section 3, we prove that languages in L_2 have representing polynomials with sub-quadratic-length witnesses. In Section 4, we present a methodology that allows to apply our HVSZK arguments-of-knowledge together with homomorphic cryptosystems to a variety of cryptographic protocols. Finally, the appendix describes our simplifications and extensions to the Damgård-Fujisaki commitment scheme together with a new and efficient argument system for nonnegativity.

2 Preliminaries and Notation

We say that an algorithm f is *efficient* when f works in the probabilistic polynomial time with respect to the summatory length of its parameters; we denote the set of efficient algorithms by \mathcal{EA} . Let $\text{bit}(x, i)$ denote the i th bit of x , i.e., $x = \sum_{i \geq 0} \text{bit}(x, i) \cdot 2^i$. When D is a distribution (including the output distribution of some probabilistic algorithm) then $x \leftarrow D$ denotes the choice of a random element x according to D . We denote the uniform distribution over a set S also by S ; that is, $x \leftarrow S$ means that x is chosen uniformly and randomly from S .

Bounded Arithmetic. Bounded arithmetic is a first-order theory of the natural numbers with non-logical symbols $0, \sigma, +, \cdot, \leq, \dot{+}, \lfloor x/2 \rfloor, |x|, \text{MSP}(x, i)$ and \sharp . The symbols $0, \sigma(x) := x + 1, +, \cdot$, and \leq have their usual meaning. Other operations are defined as $x \dot{+} y := \max(x - y, 0)$, $|x| := \lfloor \log_2(x + 1) \rfloor$, $\text{MSP}(x, i) := \lfloor x/2^i \rfloor$ and $x \sharp y := 2^{|x| \cdot |y|}$. For our purposes we adapt a slightly modified definition of bounded arithmetic where the underlying domain is \mathbb{Z} instead of \mathbb{N} . We denote by L_2 the set of terms of the quantifier-free bounded arithmetic (over \mathbb{Z}).

One can express a large number of relations in L_2 . Many familiar predicates (like $[\mu_1 > \mu_2]$, $[\mu \text{ is a perfect square}]$, $[\mu_2 = \text{bit}(\mu_1, i)]$) are known to belong to L_2 . They can be readily found from the literature.

Lucas Sequences. All nonnegative integral solutions (x, y) of the equation $x^2 - axy - y^2 = 1$ are either equal to $(Z_a(n + 1), Z_a(n))$ or $(Z_a(n), Z_a(n + 1))$,

$n \geq 0$, where $Z_a(n)$ (that we mostly denote by $a^{[n]}$) can be computed by using the next recurrent identities [Mat93]: $Z_a(0) := 0$, $Z_a(1) := 1$, and $Z_a(n+2) := aZ_a(n+1) - Z_a(n)$ for $n \geq 0$. Thus, $\{Z_a(n)\}_{n \in \mathbb{N}}$ is a Lucas sequence. Another important property of $Z_a(n)$ is that when $a > 2$ and $n > 0$ then $(a-1)^n \leq Z_a(n+1) \leq a^n$. The next variant of the Russian peasant algorithm can be used to efficiently compute the pair $(Z_a(n+1), Z_a(n))$:

Lemma 1. *The next algorithm computes $(Z_a(n+1), Z_a(n))$ from (a, n) by doing $\approx 3 \cdot \log_2 n$ two-variable multiplications in average:*

1. $\ell := \lfloor \log_2 n \rfloor$; $z := 1$; $z' := 0$
2. **for** $i := \ell$ **downto** 0 **do**
 - $t := z$; **if** $\text{bit}(n, i) = 1$ **then** $z := z(at - 2z')$; $z' = t^2 - z'^2$
else $z := t^2 - z'^2$; $z' = z'(2t - az')$;
3. Return (z, z') .

Proof. Follows from the identities $Z_a(2n) = Z_a(n)(2Z_a(n+1) - aZ_a(n)) = Z_a(n)(aZ_a(n) - 2Z_a(n-1))$ and $Z_a(2n+1) = Z_a^2(n+1) - Z_a^2(n)$. \square

While a similar $O(\log n)$ -time algorithm for Lucas sequences is described, for example, in [JQ96], the algorithm presented there works for somewhat different sequences and requires $4.5(\log_2 n + O(1))$ multiplications. Log-time algorithms for Lucas sequences have been known at least since [Wil82].

Arguments of Knowledge. For bit-strings a and μ , and predicate $Q(\cdot)$, we denote by $\text{AK}(Q(a, \mu))$ a three-round honest-verifier statistical zero-knowledge (HVSZK) two-party argument of knowledge (AoK) that given a value a (known to both parties), the arguer knows an integer parameter μ , such that the predicate $Q(a, \mu)$ is true. We always denote the values, knowledge of which has to be proved, by Greek letters; the scope of such variables lies within a single AoK. The symbol ω will always denote an auxiliary witness. As an example, $\text{AK}(y = E_K(\mu; \rho) \wedge \mu^2 = \omega)$ denotes a HVSZK AoK that given a ciphertext y and a public key K , the arguer knows a plaintext μ and a randomness ρ such that $y = E_K(\mu; \rho)$ and μ is a perfect square. Our protocols will be AoK-s in the model of Damgård and Fujisaki [DF02]. An important property of the zero-knowledge arguments is that the verifier cannot extract (significant) additional information even if he is given infinite time. This makes AoK-s more attractive than proofs of knowledge in applications where privacy of the arguer is paramount. A HVSZK argument system can be made non-interactive by using the Fiat-Shamir heuristic [FS86] in the random-oracle model. The converted argument is also secure against malicious verifiers. There exist alternative methods for converting a HVSZK argument into a full interactive zero-knowledge argument that do not use random oracles. For the purpose of Fiat-Shamir heuristic, we introduce a random oracle $H : \{0, 1\}^* \rightarrow \{0, 1\}^{2k}$.

Integer Commitment Schemes. A *secure* (in the sense of being statistically hiding and computationally binding) *integer commitment scheme* C allows the arguer $A \in \mathcal{EA}$ to commit to an integer $m \in \mathbb{Z}$, so that (1) for uniform and random r_1, r_2 and any $m_1, m_2 \in \mathbb{Z}$, the distributions $C_K(m_1; r_1)$ and $C_K(m_2; r_2)$

are statistically close; and (2) it is intractable for A to find m_1, m_2, r_1 and r_2 , such that $m_1 \neq m_2$ but $C_K(m_1; r_1) = C_K(m_2; r_2)$. Known integer commitment schemes include [FO99, DF02]; the security of both integer commitment schemes bases on some reasonable security assumptions that seem to be satisfied by class groups and a large variety of the RSA groups. We will give a description of a simplified Damgård-Fujisaki scheme in Appendix A. The main simplifications are: (a) In revealing phase, it is sufficient for the committer to send the pair (m, r) instead of the triple (m, r, b) and (b) The underlying root assumption is modified to have the following, simpler, form: given random y , it is hard to produce such (x, d, e) that $y^e = x^{de}$ and e is reasonably small.

By using a secure integer commitment scheme, one can build an HVSZK argument system for different relations between committed integers μ_i . In all such argument systems, arguer and verifier have to fix, for every i , an a priori upper bound M_i to input μ_i [FO99, DF02]. The argument system is guaranteed to have the statistical zero-knowledge property only if $|\mu_i| < M_i$. Therefore, in such protocols the interaction length depends on $\log_2 M_i$, and thus it is beneficial to precompute as precise values of M_i as feasible. Certainly it must be the case that $\log_2 M_i = k^{O(1)}$. Additionally, we will describe in Appendix A how to commit to an integer tuple (and not just to an integer). The resulting *integer tuple commitment scheme* can be used to construct more efficient arguments of knowledge than the Damgård-Fujisaki commitment scheme by itself.

Diophantine Complexity. Based on the earlier work of Davis, Putnam and Robinson, Matiyasevich proved in 1970 [Mat70] that every recursively enumerable set is Diophantine (this important result is known as the DPRM theorem), solving thus negatively Hilbert's tenth problem from year 1900. This work on the Hilbert's tenth problem has had many interesting consequences. See [Mat93] for a representation of main results of this work and related history. In 1976, Adleman and Manders [AM76] proposed the next complexity-theoretic class **D** of sets: $S \in \mathbf{D}$ iff there exists a *representing polynomial* \mathfrak{R}_S , such that $\mu \in S \iff (\exists \omega)[|\sum_i \omega_j| = |\sum_i \mu_i|^{O(1)} \wedge \mathfrak{R}_S(\mu; \omega) = 0]$. Obviously, $\mathbf{D} \subseteq \mathbf{NP}$. On the other hand, Adleman and Manders showed that several **NP**-complete problems belong to the class **D** and, based on that, conjectured that $\mathbf{D} = \mathbf{NP}$. Their conjecture was later implicitly supported by Jones and Matiyasevich [JM84] who proved that $\mathbf{D} = \mathbf{NP}$ iff the set $\{(\mu_1, \mu_2) : \mu_1 \leq_2 \mu_2\}$ belongs to **D** (Here, $\mu_1 \leq_2 \mu_2$ iff $\text{bit}(\mu_1, i) \leq \text{bit}(\mu_2, i)$ for every i .) and by Pollet [Pol03], who recently showed that when $\mathbf{co-NLOGTIME} \subseteq \mathbf{D}$ then $\mathbf{D} = \mathbf{NP}$. The gap between **co-NLOGTIME** and **NP** is wide and thus, as expected, not much is known about the actual power of the class **D**.

In the following, let M_i be some a priori upper bound on the length of the input μ_i and let W_j be a similar upper bound on the witness ω_j that holds when the lengths of the input μ_i never exceed the values M_i . Let $M := \max_i M_i$ and $W := \max_j W_j$; note that the value W is a function of M and \mathfrak{R}_S . Since the number of witnesses m and the degree of the polynomial \mathfrak{R}_S do not depend on the input size M , the total size of inputs to the representing polynomial will be $\Theta(M + W)$. Now, $S \in \mathbf{D}$ if for some representing polynomial \mathfrak{R}_S , $W = M^{O(1)}$

and therefore, the Adleman-Manders conjecture says that $S \in \mathbf{NP}$ iff for some polynomial \mathfrak{R}_S , $\mu \in S \iff (\exists \omega)[\mathfrak{R}_S(\mu; \omega) = 0 \wedge W = M^{O(1)}]$.

In the standard definition of Diophantine sets [Mat93] only nonnegative witnesses are admitted. The classes **D** and **PD** do not change when we modify their definitions to allow negative integer witnesses, since $\mu \in S \iff (\exists \omega, \omega' \in \mathbb{N}_0^m)[\mathfrak{R}_S(\mu; \omega_1 - \omega'_1, \dots, \omega_m - \omega'_m) = 0]$. On the other hand, if S has a representing polynomial $\mathfrak{R}'_S(\mu; \omega)$ with nonnegative witnesses, then S can be represented by $\mathfrak{R}'_S(\sum_{i=1}^4 \mu_{1i}^2, \dots, \sum_{i=1}^4 \mu_{ni}^2; \sum_{i=1}^4 \omega_{1i}^2, \dots, \sum_{i=1}^4 \omega_{mi}^2)$; the latter follows from a classical theorem of Lagrange (see also Thm. 2). For convenience, we will implicitly assume that all the variables belong to \mathbb{Z} (and not to \mathbb{N}_0).

3 Bounded Arithmetic Is in PD

First, let us introduce a new complexity class **PD** that is a Diophantine analogue of **P**. Namely, we say that $S \in \mathbf{PD}$ iff there is a polynomial $\mathfrak{R}_S \in \mathbb{Z}[X]$, such that (1) there exists an efficient *witness algorithm* $\mathfrak{P}_S \in \mathcal{EA}$, such that if $\mu \in S$ then $\mathfrak{R}_S(\mu; \mathfrak{P}_S(\mu)) = 0$; (2) if $\mu \notin S$ then for any ω with $|\omega| = |\mu|^{O(1)}$, $\mathfrak{R}_S(\mu; \omega) \neq 0$. Recently, Pollett proved that all sets in L_2 belong to **D** [Pol03]. We extend this to a proof that all sets in L_2 belong to **PD**.

Theorem 1. *All L_2 -terms belong to **PD**, with $W = M^{2-\varepsilon}$ for $\varepsilon > 0$.*

Proof. To show that L_2 -terms belong to **PD**, we will first show that all non-logical basic relations of bounded arithmetic belong to **PD**. Thereafter, we show how to implement the Boolean operators that connect them by using induction on the structure of formulas. Clearly, the first four basic non-logical symbols (0, σ , $+$, \cdot) have representing polynomials with no auxiliary witnesses. (For example, the predicate $[\mu_2 = \sigma(\mu_1)]$ is represented by the polynomial $\mathfrak{R}_S(\mu_1, \mu_2) = \mu_2 - \mu_1 - 1$.) The representing polynomial for \leq can be constructed by using the representing polynomial for non-negativity, see Thm. 2.

The Boolean operators \wedge , \vee and \neg can be dealt with as follows. Let $S, S' \in \mathbf{PD}$ have representing polynomials \mathfrak{R}_S and $\mathfrak{R}_{S'}$ and witness algorithms \mathfrak{P}_S and $\mathfrak{P}_{S'}$. Then $\mathfrak{R}_{S \cup S'}(\mu; \omega, \omega') = \mathfrak{R}_S(\mu; \omega) \cdot \mathfrak{R}_{S'}(\mu; \omega')$, $\mathfrak{R}_{S \cap S'}(\mu; \omega, \omega') = \mathfrak{R}_S(\mu; \omega)^2 + \mathfrak{R}_{S'}(\mu; \omega')^2$ and $\mathfrak{P}_{S \cup S'}(\mu) = \mathfrak{P}_{S \cap S'}(\mu) = (\mathfrak{P}_S(\mu), \mathfrak{P}_{S'}(\mu))$. Therefore, if $S_1 \in \mathbf{X}$ then also $S_1 \cup S_2, S_1 \cap S_2 \in \mathbf{X}$ for $\mathbf{X} \in \{\mathbf{D}, \mathbf{PD}\}$. One can establish that $\neg P(\cdot)$ belongs to **PD** by induction, assuming that $P(\cdot)$ belongs to **PD** and then studying the case of every possible main connective of P separately. (This can introduce some new witnesses.) As an example, $[\mu_1 \neq \mu_2] \equiv [(\mu_1 < \mu_2) \vee (\mu_2 > \mu_1)]$.

Three of the remaining operations can now be defined as $[\mu_3 = \mu_1 \dot{-} \mu_2] \equiv [((\mu_1 - \mu_2 = \mu_3) \wedge (\mu_1 \geq \mu_2)) \vee (\mu_3 = 0 \wedge \mu_1 < \mu_2)]$, $[\mu_2 = \lfloor \mu_1 / 2 \rfloor] \equiv [(\mu_1 = 2\omega_1) \vee (\mu_1 = 2\omega_1 + 1)]$ and $[\mu_2 = \mathbf{MSP}(\mu_1, i)] \equiv [(\mu_1 = 2^i \cdot \mu_2 + \omega \wedge \omega \in [0, 2^i - 1])]$. Note that only the last three operations need a nonempty witness ω , with $W = O(M)$. That $[\mu_3 = \mu_1^{\mu_3}]$ is in **PD** follows from Thm. 3. Finally, $[\mu_2 = |\mu_1|] \equiv [\omega_1 = 2^{\mu_2} \wedge \omega_1 \leq 2(\mu_1 + 1) \wedge (\mu_1 + 1) < \omega_1]$. Thus, $[\mu_3 = \mu_1 \# \mu_2] \equiv [(\omega_1 = |\mu_1|) \wedge (\omega_2 = |\mu_2|) \wedge (\mu_3 = 2^{\omega_1 \cdot \omega_2})]$. The theorem follows from Thm. 3,

Algorithm 1 Algorithm for computing an Lagrange representation $\mu = \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2$, $\omega \leftarrow \text{Lagrange}(\mu)$

1. Write μ in the form $\mu = 2^t(2k+1)$, where $t, k \geq 0$.
 2. If $t = 1$, then
 - (a) Choose random $\omega_1 \leq \sqrt{\mu}$, $\omega_2 \leq \sqrt{\mu - \omega_1^2}$, such that exactly one of ω_1, ω_2 is even. Let $p \leftarrow \mu - \omega_1^2 - \omega_2^2$. Now $p \equiv 1 \pmod{4}$.
 - (b) Hoping that p is prime, try to express $p = \omega_3^2 + \omega_4^2$ as follows: First, find a solution u to the equation $u^2 \equiv -1 \pmod{p}$. Apply the Euclidean algorithm to (u, p) , take the first two remainders that are less than \sqrt{p} to be ω_3 and ω_4 . If $p \neq \omega_3^2 + \omega_4^2$, p was not prime, so go back to step 2a.
 - (c) Return $(\omega_1, \dots, \omega_4)$ as the representation.
 3. If t is odd but not 1, find a representation $(\omega_1, \dots, \omega_4)$. Return $(s\omega_1, \dots, s\omega_4)$, where $s = 2^{(t-1)/2}$.
 4. If t is even, find a representation $\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2$ for $2(2k+1)$ by step 2. Then convert this to a representation for $(2k+1)$ as follows: Group $\omega_1, \omega_2, \omega_3, \omega_4$ so that $\omega_1 \equiv \omega_2 \pmod{2}$ and $\omega_3 \equiv \omega_4 \pmod{2}$. Return $(s(\omega_1 + \omega_2), s(\omega_1 - \omega_2), s(\omega_3 + \omega_4), s(\omega_3 - \omega_4))$, where $s = 2^{t/2-1}$.
-

that, together with Thm. 2, will finish this proof when we note that by induction on the length of formulas, all terms of L_2 have witnesses of sub-quadratic length, $W = M^{2-o(1)}$. \square

Next, we show that non-negativity and exponential relation have representing polynomials with sub-quadratic W . These results are novel in the following sense. First, in the proof of non-negativity we propose a slightly more efficient witness algorithm, compared to the prior art. Our system of Diophantine equations for the exponential relation, on the other hand, has substantially shorter witnesses compared to what was known previously for this relation [AM76].

Theorem 2. *An integer μ can be represented as $\mu = \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2$ with integer ω_i iff $\mu \geq 0$. Moreover, if $\mu \geq 0$ then the corresponding representation $(\omega_1, \omega_2, \omega_3, \omega_4)$ can be computed efficiently by using Algorithm 1.*

Proof. First, no negative integer is a sum of four squares. Second, if $\mu \geq 0$, μ can be decomposed as $\sum_{i=1}^4 \omega_i^2$ by a well-known result of Lagrange from 1770. Rabin and Shallit [RS86] proposed a probabilistic polynomial-time algorithm for computing the witnesses ω_i . The new Algorithm 1 is somewhat more efficient, due to the pairing of the Rabin-Shallit algorithm with the well-known Cornacchia algorithm from 1908 [Coh95, Section 1.5.2] that, given a prime $p \equiv 1 \pmod{4}$, finds a pair (ω_3, ω_4) , such that $p = \omega_3^2 + \omega_4^2$. (To compare, the original Rabin-Shallit algorithm used the full Euclidean algorithm over Gaussian integers, while Cornacchia's algorithm uses the partial Euclidean algorithm over integers). Finally, square root of -1 modulo p can be found efficiently. \square

Exponential Relation Is in PD. For a long time, finding a representing polynomial for the exponential relation was the last open issue in the solution of

the Hilbert's 10th problem [Mat93]. Matiyasevich was the first to describe an explicit representing polynomial for the exponential relation. Alternative polynomials were later found in [Dav73,JSWW76], but none of these polynomials is really practical for our purposes due to at least cubic-length witnesses. However, Adleman and Manders showed in 1976 [AM76] that when one allows exponentially long witnesses when $x \notin S$ then the polynomial proposed in [MR75] can be modified to have sub-quadratic-length witnesses when $x \in S$.

Next, we construct a new representing polynomial that is slightly more efficient than the one in [AM76]. Our proof bases on ideas from [AM76,Mat93], [Rob52]. To prove our result, we use crucially the next lemma that is an analogue of Lemma VII from [AM76]. ([AM76, Lemma VII] was stated for a different Lucas sequence, worked only when $c < 2d$, and guaranteed only that either $a < (2c)^d$ or $a \geq c^c$.)

Lemma 2. *Let (a, b, c, d) be any integers with $c > d + 2 \geq 2$. If $[(a^2 - cab - b^2 = 1) \wedge (0 \leq a < b) \wedge (a \equiv d \pmod{c-2})]$, then either $(a, b) = (c^{\lfloor d \rfloor}, c^{\lfloor d+1 \rfloor})$ and $a \leq c^{d-1}$, or $(a, b) \neq (c^{\lfloor d \rfloor}, c^{\lfloor d+1 \rfloor})$ and $a \geq (c-1)^{d+c-3}$.*

Proof. Let (a, b, c, d) be such integers. Since $[(a^2 - cab - b^2 = 1) \wedge (0 \leq a < b)]$, then $(a, b) = (c^{\lfloor x \rfloor}, c^{\lfloor x+1 \rfloor})$ for some $x \in \mathbb{N}_0$. Since $e^{\lfloor f \rfloor} \equiv f \pmod{e-2}$ for any e, f [Mat93], $[a \equiv d \pmod{c-2}]$ guarantees that $x \equiv d \pmod{c-2}$. Since $c > d + 2$, then $(a, b) = (c^{\lfloor d+k(c-2) \rfloor}, c^{\lfloor d+k(c-2)+1 \rfloor})$ for some $k \geq 0$. If $x = d$ then $a = c^{\lfloor d \rfloor} \leq c^{d-1}$. On the other hand, if $x \neq d$ then $a \geq c^{\lfloor d+(c-2) \rfloor} \geq (c-1)^{d+c-3}$. \square

Theorem 3. *Assume $\mu_1 > 1$, $\mu_3 > 0$ and $\mu_2 > 2$. The exponential relation $[\mu_3 = \mu_1^{\mu_2}]$ belongs to **PD**. More precisely, let $E(\mu_1, \mu_2, \mu_3)$ be the next equation:*

$$\begin{aligned}
 & [(\exists \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6)(\exists_b \omega_7, \omega_8)] \\
 & [(\omega_2 = \omega_1 \mu_1 - \mu_1^2 - 1) \wedge (\omega_2 - \mu_3 - 1 \geq 0) \wedge \quad (E1 - E2) \\
 & (\mu_3 - (\mu_1 - \omega_1) \omega_7 - \omega_8 = \omega_2 \omega_3)) \wedge (\omega_1 - 2 \geq 0) \wedge \quad (E3 - E4) \\
 & ((\omega_1 - 2)^2 - (\mu_1 + 2)(\omega_1 - 2) \omega_5 - \omega_5^2 = 1) \wedge \quad (E5) \\
 & (\omega_1 - 2 = \mu_2 + \omega_6(\mu_1 + 2)) \wedge (\omega_7 \geq 0) \wedge (\omega_7 < \omega_8) \wedge \quad (E6 - E8) \\
 & (\omega_7^2 - \omega_1 \omega_7 \omega_8 - \omega_8^2 = 1) \wedge (\omega_7 = \mu_2 + \omega_4(\omega_1 - 2)) \quad (E9 - E10)
 \end{aligned}$$

where “ \exists_b ” signifies a bounded quantifier in the following sense: if $\mu_3 = \mu_1^{\mu_2}$ then $E(\mu_1, \mu_2, \mu_3)$ is true with $W = \Theta(\mu_2^2 \log \mu_1) = o(M^2)$. On the other hand, if $\mu_3 \neq \mu_1^{\mu_2}$ then either $E(\mu_1, \mu_2, \mu_3)$ is false, or it is true but the intermediate witnesses ω_7 and ω_8 have length $\Omega(\mu_3 \log \mu_3)$, which is equal to $\Omega(2^M \cdot M)$ in the worst case.

(Note that 16 additional witnesses are needed in four inequalities. For the sake of simplicity we will not enlist all of them.)

Proof. Denote the i th conjunctive subformula of E by Ei . We will proceed by showing that the required witnesses are $\omega_1 \leftarrow (\mu_1 + 2)^{\lfloor \mu_2+1 \rfloor} + 2$, $\omega_2 \leftarrow \omega_1 \mu_1 -$

$\mu_1^2 - 1$, $\omega_3 \leftarrow (\mu_3 - (\mu_1 - \omega_1)\omega_1^{\llbracket \mu_2 \rrbracket} - \omega_1^{\llbracket \mu_2+1 \rrbracket})/\omega_2$, $\omega_4 \leftarrow (\omega_8 - \mu_2)/(\omega_1 - 2)$, $\omega_5 \leftarrow (\mu_1 + 2)^{\llbracket \mu_2+2 \rrbracket}$, $\omega_6 \leftarrow (\omega_1 - 2 - \mu_2)/(\mu_1 + 2)$, $\omega_7 \leftarrow \omega_1^{\llbracket \mu_2 \rrbracket}$ and $\omega_8 \leftarrow \omega_1^{\llbracket \mu_2+1 \rrbracket}$.

Really, let

$$B_a := \begin{pmatrix} a & -1 \\ 1 & 0 \end{pmatrix}, \quad \text{then} \quad B_a^r = \begin{pmatrix} a^{\llbracket r+1 \rrbracket} & -a^{\llbracket r \rrbracket} \\ a^{\llbracket r \rrbracket} & -a^{\llbracket r-1 \rrbracket} \end{pmatrix}$$

for any a and r . For an ω_1 that we will fix later, let $\omega_2 := \omega_1\mu_1 - \mu_1^2 - 1$, i.e., assume that $E1$ holds. Then, $(\mu_1, 1)^\top$ is an eigenvector of B_{ω_1} modulo ω_2 , with eigenvalue μ_1 , since $B_{\omega_1} \cdot (\mu_1, 1)^\top = (\omega_1\mu_1 - 1, \mu_1)^\top \equiv (\mu_1^2, \mu_1)^\top = \mu_1 \cdot (\mu_1, 1)^\top \pmod{\omega_2}$. Therefore,

$$\begin{pmatrix} \omega_1^{\llbracket \mu_2+1 \rrbracket} & -\omega_1^{\llbracket \mu_2 \rrbracket} \\ \omega_1^{\llbracket \mu_2 \rrbracket} & -\omega_1^{\llbracket \mu_2-1 \rrbracket} \end{pmatrix} \cdot \begin{pmatrix} \mu_1 \\ 1 \end{pmatrix} = B_{\omega_1}^{\mu_2} \cdot \begin{pmatrix} \mu_1 \\ 1 \end{pmatrix} \equiv \mu_1^{\mu_2} \cdot \begin{pmatrix} \mu_1 \\ 1 \end{pmatrix} \pmod{\omega_2}.$$

In particular, $\mu_1\omega_1^{\llbracket \mu_2 \rrbracket} - \omega_1^{\llbracket \mu_2-1 \rrbracket} \equiv \mu_1^{\mu_2} \pmod{\omega_2}$. Now, as soon as $\mu_1^{\mu_2} < \omega_2$, we can write $[\mu_3 = \mu_1^{\mu_2}] \iff [E2 \wedge (\mu_1\omega_1^{\llbracket \mu_2 \rrbracket} - \omega_1^{\llbracket \mu_2-1 \rrbracket} \equiv \mu_1^{\mu_2} \pmod{\omega_2})]$.

One can guarantee that $\mu_1^{\mu_2} < \omega_2$ by selecting ω_1 , so that $\omega_1 \geq \mu_1^{\mu_2-1} + \mu_1 + 2$. To be able later to apply Lemma 2, it also must be the case that $\omega_2 > \mu_2 + 2$. Since $\mu_1 > 1$, we can choose $\omega_1 \leftarrow (\mu_1 + 2)^{\llbracket \mu_2 \rrbracket} + 2 \geq (\mu_1 + 1)^{\mu_2-1} + 2 \geq \mu_1^{\mu_2-1} + \mu_1 + 2$. Since $\mu_1 > 0$, we can invoke Lemma 2 with $(a, b, c, d) = (\omega_1 - 2, \omega_5, \mu_1 + 2, \mu_2)$. Since here it suffices to show that $\omega_1 - 2 = (\mu_1 + 2)^{\llbracket \mu_2+k\mu_1 \rrbracket}$ and $\omega_5 = (\mu_1 + 2)^{\llbracket \mu_2+k\mu_1+1 \rrbracket}$ for some $k > 0$, we are done by adding two verifications ($E5$ and $E6$) from Lemma 2. (More precisely, here we one does not have to verify that $\omega_1 - 2 < \omega_5$.)

Now, due to the choice of ω_1 , $\omega_1 > (\mu_1 + 1)^{\mu_2-1} + 2 \geq \mu_2 + 2$. Therefore, Lemma 2 with inputs $(a, b, c, d) = (\omega_7, \omega_8, \omega_1, \mu_2)$ guarantees that after doing the verifications ($E7 - E10$), one can be assured that one of the next two cases is true. First, $(\omega_7, \omega_8) = (\omega_1^{\llbracket \mu_2 \rrbracket}, \omega_1^{\llbracket \mu_2+1 \rrbracket})$. Then $|\omega_7| \approx |\omega_8| \approx \mu_2 \cdot |\omega_1| \approx \mu_2^2 \cdot |\mu_1| \leq \mu_2^2 \cdot |\mu_1| < |M_3|^2 < 2|M|^2$. (Note that $M \approx \mu_2|\mu_1|$.) Second, $(\omega_7, \omega_8) \neq (\omega_1^{\llbracket \mu_2 \rrbracket}, \omega_1^{\llbracket \mu_2+1 \rrbracket})$, but then $|\omega_7| \geq |(\omega_1 - 1)^{\omega_1-2}| \approx \omega_1|\omega_1| \approx \mu_1^{\mu_2-1} \cdot \log_2 \mu_1^{\mu_2-1} \geq \mu_3 \cdot \log_2 \mu_3 \approx 2^M \cdot M$, which is exponential in the input size. \square

The largest Z -function occurring in this lemma is

$$\omega_8 = \omega_1^{\llbracket \mu_2+1 \rrbracket} = Z_{(\mu_1+2)^{\llbracket \mu_2 \rrbracket}+2}(\mu_2 + 1) \leq Z_{(\mu_1+2)^{\mu_2-1}}(\mu_2 + 1) \leq (\mu_1 + 2)^{\mu_2^2 - \mu_2}.$$

For comparison, [AM76] used an equation system from [MR75], where the largest ψ -function (for a different Lucas sequence ψ) is $\psi_{4\mu_2\mu_1(\mu_3+1)+\mu_1^2+2\mu_1}(\mu_2 + 1)$.

The cases $\mu_1 \in [0, 1]$, $\mu_3 = 0$ and $\mu_2 \in [0, 1, 2]$ can be handled trivially, and therefore the exponential relation belongs to **PD** for any μ_1, μ_2, μ_3 . One application of this theorem is that an arbitrary Turing machine can be emulated by a slightly more efficient Diophantine Turing machine than it was known before [AM76].

4 Cryptographic Applications

Diophantine Membership Arguments. Given a secure integer commitment scheme with efficient HVSZK AoK-s for additive and multiplicative relations, one can argue in HVSZK that any polynomial relation holds between a tuple of committed integers [FO99]. That is, one can argue in HVSZK that $p(\mu) = 0$ for some fixed $p \in \mathbb{Z}[X]$, and a committed $\mu \in \mathbb{Z}^n$.

We will expand the [FO99]-methodology as follows. When $S \in \mathbf{D}$ and the arguer knows the witness, then by using an integer commitment scheme, she can argue in HVSZK that she knows an auxiliary (suitably chosen) witness ω , such that $\mathfrak{R}_S(\mu; \omega) = 0$, where \mathfrak{R}_S is again the representing polynomial of S . This results in a what we call a *Diophantine argument system* $\text{AK}(c_1 = C_K(\mu_1, \dots, \mu_n; \rho_1) \wedge (\mu_1, \dots, \mu_n) \in S)$.

The asymptotical communication complexity of the resulting Diophantine argument system is $\Theta(W + M)$, where the constant depends on the number of parameters and witnesses, but also on the degree of \mathfrak{R}_S and on the internal structure of \mathfrak{R}_S . (For example, a Diophantine argument system for $\mu_1 + \mu_2 = \omega_1^4 + \omega_2^2$ requires a constant times more interaction than the one for $\mu_1 = \omega_1^2$.) Thus, Diophantine argument systems with interaction $M^{O(1)}$ exist for all $S \in \mathbf{D}$. In particular, an immediate corollary of the positive solution to the Adleman-Manders conjecture $\mathbf{NP} = \mathbf{D}$ is that every set $S \in \mathbf{NP}$ has a Diophantine HVSZK argument system with communication complexity $M^{O(1)}$. However, there are two practical considerations.

First, if (say) $W = M^{\Omega(2)}$ then the resulting argument systems are asymptotically too long to have immediate applications in cryptography. As we also so in this paper, finding representing polynomials \mathfrak{R}_S with small W is a nontrivial task, and it often needs breakthroughs in number theory.

Note also that quadratic length seems to be a reasonable metering point, since for many interesting predicates one can build trivial quadratic-length zero-knowledge arguments (here and in the following, assume for the sake of simplicity that the input length M is larger than the security parameter k). In such AoK-s, one separately commits to every bit of the input, and then shows that the committed bits satisfy some Boolean formula. An immediate corollary of Theorem 1 is that one can build *sub-quadratic-length* HVSZK AoK-s for all languages from L_2 . Therefore, our AoK-s are an improvement upon such argument systems.

Second, if $S \in \mathbf{D} \setminus \mathbf{PD}$, the arguer cannot efficiently find the witness ω for every relevant input μ . In such a case, the witness ω can be seen as a trap-door information. However, this case is still relevant in certain cryptographic applications. For example, the relation $[\mu \text{ is composite}] \equiv [(\exists y_1, y_2 \leq \mu)[\mu = y_1 y_2 \wedge y_1 > 1 \wedge y_2 > 1]]$ does not have a witness algorithm, given that factoring is hard. (The resulting argument system that a committed number is composite can be compared to a more complex protocol by Poupard and Stern [PS00].) In particular this means that $\mathbf{D} \neq \mathbf{PD}$, unless factoring is easy.

Note that to apply the previously described methodology, one needs to both encrypt and commit all messages. Additionally, one needs to argue that that encrypted and committed messages are equal. This can be done straightforwardly

by using standard cryptographic tools. We finish the paper with concrete applications and protocols. There are definitely more applications than we mention in the following. In particular, our methodology is not limited to the outsourcing model.

Example: Efficient Range Proofs. A cryptographically important argument system for \mathbb{N}_0 (a partial list of potential applications to this argument system can be found in [Bou00], it includes electronic cash systems, verifiable encryption, group signatures, publicly verifiable secret sharing schemes and other zero-knowledge protocols; more applications can be found in [LAN02] and in the current paper) can be based on Theorem 2. Briefly, during this argument system, the arguer first represents μ as $\mu = \omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2$ (here, $\omega = (\omega_1, \dots, \omega_4)$ is the witness). After that, she argues in HVSZK that she knows such a representation. Our argument system bases on the new integer tuple commitment scheme. The full argument system is described in Appendix B. A non-interactive version of such argument system is ≈ 1700 bytes long for realistic security parameters. This is slightly shorter than Boudot's argument system [Bou00] for the same problem. Additionally, our argument system is perfectly complete, while Boudot's argument system is not. A nice demonstration of the usefulness of the new integer tuple commitment scheme (presented in Appendix A) is the fact that this argument system has only ≈ 1.9 times larger non-interactive argument than the original multiplication proof of Damgård and Fujisaki; this is achieved by doing four squarings in parallel.

Outsourcing Model. A general setting in many cryptographic protocols (like voting and auctions [LAN02]) involves a set of participants, an authority and possibly an impartial third party. The participants make social or financial choices $\{v_i\}$, encode them as $\{\text{enc}(v_i)\}$ by using some encoding function enc , and then encrypt the resulting encodings by using a homomorphic public-key cryptosystem and third party's public key, and send the results, together with an HVSZK argument of correctness, to the authority. (Of course, we assume that all the steps are authenticated.) The authority multiplies the ciphertexts and sends the product $\prod_i E_K(\text{enc}_i(v_i)) = E_K(\sum_i \text{enc}(v_i))$ to the third party. The third party decrypts the result, obtains the sum $\sum_i \text{enc}(v_i)$ and applies a decoding function dec to obtain the vector $e = (\dots, e_j, \dots)$, where e_j can for example be the number of voters whose choice was j . The third party applies some function final to e , and sends $\text{final}(e)$ to the authority together with an zero-knowledge argument of correctness that $\text{final}(e)$ was correctly computed. The authority then broadcasts $\text{final}(e)$ and the argument of correctness to all participants.

As an example, final could be an identity function. Then this model will implement a common voting process with an accountable third party. If $\text{final}(e) = j_0$ where $e_{j_0} = \max e_j$, one could implement voting with minimal information disclosure. Namely, the authority would only get to know the name of the winner. To the best of our knowledge, there are no such efficient prior art voting schemes. One can also implement the $(b+1)$ st-price auctions by choosing $\text{final}(e) = j_0$, where j_0 is the $(b+1)$ st largest social choice [LAN02]. (This includes Vickrey auctions, for example.)

In general, the “outsourcing” model enables one to construct secure and extremely efficient voting (or auction) schemes with the only drawback that the third party (but only she) will get to know the value of e . In particular, this enables one to avoid threshold trust. See [LAN02] for a discussion why at least in the auction scenario, the information leakage to the authority does not matter but the property of not using threshold trust does. In the most common in the real-world voting scenario, the vector e is meant to be leaked. Moreover, even in the nation-wide elections, one does not really want to have threshold trust between computers. Instead, it seems to be desirable—as show discussions with the members of electorate committees—that the encoded and encrypted vector e can be decrypted by using a single hardware-protected private key that can be used only by the presence of several trusted entities and independent experts, and will be destroyed as soon as some allocated period at the end of elections (and all election-related legal discussions) have ended.

Now, **final** can be any function for which the predicate $[y = \text{final}(x)]$ belongs to **PD**. As we have shown, extremely efficient arguments are available when $\text{final} \in L_2$. It is not known how to implement as efficiently so many different schemes for such a broad variety of functions **final** in the model that involves threshold trust but no third party like in [CGS97,DJ01]. In particular, no really efficient $(b + 1)$ st-price auctions are known in the threshold trust scenario.

Efficient Range Arguments in Exponents. The costliest part of the otherwise efficient Damgård-Jurik voting protocol from [DJ01] involves an argument for $\text{AK}(y = E_K(\text{enc}(\mu)) \wedge \mu \in [0, h])$ that is necessary to show that the votes were encoded properly. We call this argument a *range argument in exponents* (RAIE). An RAIE is also necessary in the auction protocol of [LAN02], both to show that the bids were encoded correctly, and that the authority returns the correct value of $\text{final}(e)$. The proposed AoK-s from [DJ01,LAN02] have interaction $\Theta(\max(k, m \cdot \log a) \cdot \log m) \Theta(m \cdot \log a \cdot \log m)$, where a is an a priori fixed upper bound to the number of participants, and m is the number of possible social choices. (This follows from [LAN02, Section 8], when we assume that the security parameter is approximately equal to $m \log a$.)

The most efficient known RAIE [LAN02] has $\text{enc}(\mu) := (\text{nextprime}(a))^\mu$ (where $\text{nextprime}(a)$ is the smallest prime $\geq a$) and results in a HVSZK AoK with interaction length $\Theta(m \cdot \log a)$. We propose two different RAIE-s that do not require computing the **nextprime** function. The first approach sets $\text{enc}(\mu) := Z_a(\mu + 1)$, where $Z_a(\mu)$ is the μ th element in the familiar Lucas sequence, and results in a HVSZK AoK with interaction length $\Theta(m \cdot \log a)$. Application of Z instead of the exponentiation enables us to improve over the communication efficiency of the Damgård-Jurik multi-candidate voting scheme [DJ01] and over the Lipmaa-Asokan-Niemi $(b + 1)$ st-price auction scheme [LAN02] by a factor of $\Theta(\log m)$. Finally, we propose a *Diophantine* RAIE with $\text{enc}(\mu) := a^\mu$ and interaction $\Theta(W + M) = \Theta(M^{2-\varepsilon}) = \Theta((m \cdot \log a)^{2-\varepsilon})$.

First Approach: Lucas Sequences. The function $a^{[n]} = Z_a(n)$ is a suitable replacement for exponentiation in the sense, intended in [DJ01,LAN02], since $(a - 1)^n \leq Z_a(n) \leq a^n$ whenever $a > 2$ (This makes the constants e_j in the

sum $\sum_{i=1}^a Z_{a+1}(v_i) = \sum_j e_j Z_{a+1}(j)$ unambiguous whenever $v_i \in [1, h]$, and thus makes it possible to uniquely recover the vector e from $\sum \text{enc}(e_i)$. However, we must make the plausible assumption that $a > 2$, for $a = 2$ one has to use another approach.), and that $Z_a(n)$ can be computed in time $O(\log n)$. Most importantly, one can very efficiently argue that the committed number μ belongs to the set $\{a^{\lfloor n \rfloor} : n \geq 0 \wedge n = k^{O(1)}\}$ by using the representing polynomial $\Re_S(\mu; \omega) = \omega^2 - a\mu\omega - \mu^2 - 1$. This must be accompanied by an AoK that $\mu \in [l, h]$. The length of a non-interactive version of this argument is ≈ 1200 bytes for realistic security parameters. A minor drawback of this solution is that computing $Z_a(n)$ requires about twice more resources than computing of a^n without the function `nextprime`. (Also, in some solutions one cannot readily substitute exponentiation with the function Z .) Note also that $Z_a(n)$ is not the unique Lucas sequence that satisfies all these conditions.

Second Approach. Here, one would have $\text{enc}(n) = a^n$, as in [DJ01, LAN02]. The argument system from Thm. 2 is usually not more communication-efficient than the protocols from [DJ01, LAN02], however, it is constant-round, which may have advantages in some concrete applications. (Precise analysis omitted due to the space constraints. Note that here we have the relation $[\mu_2 = a^{\mu_1}]$ for a constant a , that allows us to improve on Thm. 3.)

Acknowledgements and Further Work

This work was partially supported by the Finnish Defense Forces Research Institute of Technology. We would like to thank Yuri Matiyasevich, Jeffrey Shallit and anonymous referees for useful comments. This paper obsoletes an earlier technical report [Lip01].

Efficient Diophantine membership arguments can be given for many interesting sets $S \subset \mathbb{Z}$. We did certainly not mention all cryptographically relevant sets S that have such arguments, and the class L_2 can be certainly broadened. We hope that this paper stimulates the research both in finding more efficient representing polynomials for concrete sets S but also in giving a (positive or negative) answer to the conjecture $\mathbf{NP} = \mathbf{D}$.

References

- AM76. Leonard M. Adleman and Kenneth L. Manders. Diophantine Complexity. In *17th Annual Symposium on Foundations of Computer Science*, pages 81–88, Houston, Texas, USA, 25–27 October 1976. IEEE Computer Society Press.
- Bou00. Fabrice Boudot. Efficient Proofs that a Committed Number Lies in an Interval. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444, Bruges, Belgium, May 14–18 2000. Springer-Verlag. ISBN 3-540-67517-5.
- Bra97. Stefan Brands. Rapid Demonstration of Linear Relations Connected by Boolean Operators. In Fumy [Fum97], pages 318–333.

- CGS97. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In Fumy [Fum97], pages 103–118.
- Coh95. Henri Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1995.
- Dav73. Martin Davis. Hilbert’s Tenth Problem is Unsolvable. *American Mathematical Monthly*, 80(3):233–269, March 1973.
- DF02. Ivan Damgård and Eiichiro Fujisaki. An Integer Commitment Scheme Based on Groups with Hidden Order. In Yuliang Zheng, editor, *Advances on Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142, Queenstown, New Zealand, December 1–5 2002. Springer-Verlag.
- DJ01. Ivan Damgård and Mads Jurik. A Generalisation, a Simplification and Some Applications of Paillier’s Probabilistic Public-Key System. In Kwangjo Kim, editor, *Public Key Cryptography ’2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, Korea, 13–15 February 2001. Springer-Verlag.
- FO99. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical Zero-Knowledge Protocols to Prove Modular Polynomial Relations. *IEICE Transaction of Fundamentals of Electronic Communications and Computer Science*, E82-A(1):81–92, January 1999.
- FS86. Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology—CRYPTO ’86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, California, USA, 11–15 August 1986. Springer-Verlag, 1987.
- Fum97. Walter Fumy, editor. *Advances in Cryptology — EUROCRYPT ’97*, volume 1233 of *Lecture Notes in Computer Science*, Konstanz, Germany, 11–15 May 1997. Springer-Verlag.
- JM84. J. P. Jones and Yuri Matiyasevich. Register Machine Proof of the Theorem on Exponential Diophantine Representation of Enumerable Sets. *Journal of Symbolic Logic*, 49:818–829, 1984.
- JQ96. Marc Joye and Jean-Jacques Quisquater. Efficient Computation of Full Lucas Sequences. *Electronics Letters*, 32(6):537–538, March 1996.
- JSWW76. James P. Jones, Daihachiro Sato, Hideo Wada, and Douglas Wiens. Diophantine Representation of the Set of Prime Numbers. *American Mathematical Monthly*, 83(6):449–464, June–July 1976.
- LAN02. Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure Vickrey Auctions without Threshold Trust. In Matt Blaze, editor, *Financial Cryptography — Sixth International Conference*, volume 2357 of *Lecture Notes in Computer Science*, pages 87–101, Southampton Beach, Bermuda, March 11–14 2002. Springer-Verlag.
- Lip01. Helger Lipmaa. Statistical Zero-Knowledge Proofs from Diophantine Equations. Cryptology ePrint Archive, Report 2001/086, November 20 2001. <http://eprint.iacr.org/>.
- Mat70. Yuri Matiyasevich. Enumerable Sets are Diophantine. *Soviet Math., Doklady*, 11:354–358, 1970. English translation.
- Mat93. Yuri Matiyasevich. *Hilbert’s Tenth Problem*. Foundations of Computing. MIT Press, October 1993. ISBN 0-262-13295-8.

- MR75. Yuri Matiyasevich and Julia Robinson. Reduction of an Arbitrary Diophantine Equation to One in 13 Unknowns. *Acta Arithmetica*, 27:521–553, 1975.
- Pol03. Chris Pollett. On the Bounded Version of Hilbert’s Tenth Problem. *Archive for Mathematical Logic*, 42(5):469–488, 2003.
- PS00. Guillaume Poupard and Jacques Stern. Short Proofs of Knowledge for Factoring. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography ’2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 147–166, Melbourne, Victoria, Australia, 18–20 January 2000. Springer-Verlag.
- Rob52. Julia Robinson. Existential Definability in Arithmetic. *Transactions of the American Mathematical Society*, 72(3):437–449, May 1952.
- RS86. Michael O. Rabin and Jeffrey O. Shallit. Randomized Algorithms in Number Theory. *Communications in Pure and Applied Mathematics*, 39:239–256, 1986.
- Wil82. Hugh C. Williams. A $p + 1$ Method of Factoring. *Mathematics of Computation*, 39:225–234, 1982.

A Extensions to Damgård-Fujisaki Integer Commitment Scheme

Let $Gen \in \mathcal{EA}$ be a group generation algorithm that on the input 1^k outputs the description $\text{descr}(\mathcal{G})$ of a finite Abelian group \mathcal{G} . Apart from the usual assumptions (given $D \in \Sigma^*$, it is easy to verify that $D \in Gen(1^k)$, easy to verify whether some μ belongs to \mathcal{G} for which $D = \text{descr}(\mathcal{G})$, and easy to perform group operations in \mathcal{G} for which $D = \text{descr}(\mathcal{G})$), we require a few additional assumptions.

First, one assumes that while the arguer knows a reasonably close upper bound $2^B > \text{ord}(\mathcal{G})$ to the order of \mathcal{G} , $B = B_{\mathcal{G}}$, he does *not* know the order itself. Let $\ell(k)$ be polynomial in k . Another large number $F = F(k)$ is chosen, such that it is still feasible to factor numbers that are smaller than $F(k)$. Say, $F(k) = O(k^{\log k})$. (In our calculations we will take $F(k) = 2^{80}$ when $k = 1024$.) Based on the fundamental theorem of finite Abelian groups, one can write \mathcal{G} as $\mathcal{G} = \mathcal{U} \times \mathcal{H}$, where the order of \mathcal{U} has only prime factors at most $F(k)$ (we call such numbers $F(k)$ -smooth) and the order of \mathcal{H} has prime factors larger than $F(k)$ (we call such numbers $F(k)$ -rough).

Let $\ell(\mathcal{G}) := |\mathcal{U}|$. Then $\ell(\mathcal{G})$ is $F(k)$ -smooth. It is assumed that (1) $\ell(\mathcal{G}) \leq \ell(k)$ and that $\text{descr}(\mathcal{G})$ includes $\ell(\mathcal{G})$; (2) for any string μ it can be decided on polynomial time, based on $(x, \text{descr}(\mathcal{G}))$, whether x represents an element in \mathcal{G} . Finally, it is assumed that the next *strong divisible root assumption* holds: given a random $\mathcal{G} \leftarrow Gen(1^k)$ and $y \leftarrow \mathcal{G}$, it is hard to produce such (x, d, e) that $y^e = x^{de}$ and $e \leq \ell(\mathcal{G})$. The probability is taken over the coin tosses of Gen and of the adversary. Note that this assumption is an equivalent but simpler version of the root assumption from [DF02].

It was shown in [DF02] that \mathcal{G} can be chosen as \mathbb{Z}_n for RSA modulus $n = pq$, such that $\gcd(p-1, q-1) = 2$, $p-1$ and $q-1$ do not have too many small factors, and the strong RSA assumption holds. However, when the RSA group

\mathbb{Z}_n^* is used, one must additionally assume that the arguer does not know the value $\varphi(n)$. This may be achieved, for example, when the verifier creates n and keeps its factorisation secret.

Commitment Scheme. During the setup phase of Damgård-Fujisaki integer commitment scheme, A and V agree on the group \mathcal{G} and on a large integer $F(k)$. Verifier V chooses a random element $h \in \mathcal{G}$ (which by the group assumptions has a $F(k)$ -rough order [DF02] with an overwhelming probability. To make the order certainly $F(k)$ -rough, one might raise a random element to the power $\ell(\mathcal{G})$.) and a random secret key $s \in \mathbb{Z}_{2^{B+k}}$. V sets $g \leftarrow h^s$. Verifier V sends the public key $K = (g; h)$ to A and then proves in SZK that $g \in \langle h \rangle$. Let \mathcal{C}_{Com} denote the commitment space of the used integer commitment scheme (in this concrete case, $\mathcal{C}_{Com} = \mathcal{G}$). When committing to $m \in \mathbb{Z}$, A chooses a random $r \leftarrow \mathbb{Z}_{2^{B+k}}$ and sends $C_K(m; r) := g^m h^r$ to V . To open a commitment c , A sends to V a triple (m, r, b) , such that $c = C_K(m; r) \cdot b$ and $b^{\ell(\mathcal{G})} = 1$. (For an explanation of the role of b in the opening phase, see [DF02].) Alternatively, A can send only (m, r) to V who then verifies that $c^{\ell(\mathcal{G})} = C_K(m; r)^{\ell(\mathcal{G})}$. Clearly, this alternative is equivalent to the Damgård-Fujisaki commitment scheme in security. (The proof of this is trivial: if $c^{\ell(\mathcal{G})} = C_K(m; r)^{\ell(\mathcal{G})}$ then V can compute b as $b \leftarrow c \cdot C_K(m; r)^{-1}$. Clearly, $b^{\ell(\mathcal{G})} = 1$ and $c = C_K(m; r) \cdot b$. On the other hand, given b with $b^{\ell(\mathcal{G})} = 1$ and $c = C_K(m; r) \cdot b$, clearly $c^{\ell(\mathcal{G})} = C_K(m; r)^{\ell(\mathcal{G})}$.)

Integer Tuple Commitment Scheme. We now sketch an extension to the Damgård-Fujisaki commitment scheme that allows to simultaneously commit to a tuple of integers. As in the Damgård-Fujisaki commitment scheme, the arguer and verifier initially agree on a group \mathcal{G} , and then verifier creates a random element $h \in \mathcal{G}$. Additionally, the verifier will choose m random elements $s_i \leftarrow [0, 2^{B+k}]$, where B is a security parameter [DF02], set $g_i \leftarrow h^{s_i}$ and send the values g_i to the verifier. Apart from that, arguer A and verifier V follow the same initialisation rules as in the Damgård-Fujisaki scheme. A tuple $(\mu_1, \dots, \mu_n) \in \mathbb{Z}^n$ is committed by drawing a random integer $\rho \leftarrow [0, 2^{B+k}]$ and then setting the commitment to $C_K(\mu_1, \dots, \mu_n; \rho) := (\prod_{i=1}^n g_i^{\mu_i}) \cdot h^\rho$. During the opening phase, A sends the tuple $(\mu_1, \dots, \mu_n; \rho)$ to V , and the verifier checks that $c^{\ell(\mathcal{G})} = C_K(\mu_1, \dots, \mu_n; \rho)^{\ell(\mathcal{G})}$, where $\ell(\mathcal{G})$ is another security parameter [DF02]. (Equivalently, A can send the tuple $(\mu_1, \dots, \mu_n; \rho; b)$, and the verifier checks that $c = C_K(\mu_1, \dots, \mu_n; \rho) \cdot b$ and that $b^{\ell(\mathcal{G})} = 1$.)

It is straightforward to show that the security of the Damgård-Fujisaki integer commitment scheme and the security of the the sketched extension (that we call the RDF integer commitment scheme) are equivalent, given that the arguer does not know the mutual discrete logarithms of elements g_i . As a simple corollary, we can use the RDF integer commitment scheme C to build HVSZK AoK-s of type $\text{AK}(\dots \wedge y = C_K(\mu_1, \dots, \mu_n; \rho) \wedge \dots)$.

The RDF integer commitment scheme can be used to speed up the efficiency of many argument systems, by enabling one to prove several multiplicative or additive relations at once [Bra97]. (In contrast, without using the RDF scheme, a separate protocol must be used for every polynomial relation.) That is, such

Protocol 1 Computationally sound HVSZK argument system for the set of nonnegative integers.

1. Arguer A represents μ as $\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2$, using the algorithm from Theorem 2. For $i \in [1, 4]$, A chooses random $r_{1i} \leftarrow \mathbb{Z}_{2^{B+k}}$ such that $\sum_i r_{1i} = \rho$; A chooses random $m_{1i} \leftarrow \mathbb{Z}_{2^{kF(k)M^{1/2}}}$, $r_{2i} \leftarrow \mathbb{Z}_{2^{B+2kF(k)}}$ and lets $c_{1i} \leftarrow C_{K_i}(\omega_i; r_{1i})$. She also chooses a random $r_3 \leftarrow \mathbb{Z}_{2^{B+2kF(k)M^{1/2}}}$ and lets $c_2 \leftarrow C_{K'}(m_{11}, \dots, m_{14}; \sum_i r_{2i})$, $c_3 \leftarrow C_{(c_{11}, \dots, c_{14}; h)}(m_{11}, \dots, m_{14}; r_3)$. Arguer sends $(c_{11}, c_{12}, c_{13}, c_{14}, c_2, c_3)$ to V .
 2. V generates a random $e \leftarrow \mathbb{Z}_{F(k)}$ and sends it to A .
 3. A computes $m_{2i} = m_{1i} + e\omega_i$, $r_{4i} \leftarrow r_{2i} + e \sum r_{1i}$, $i \in [1, 4]$, and $r_5 \leftarrow r_3 + e \sum_i (1 - \omega_i)r_{1i}$. A sends $(m_{21}, m_{22}, m_{23}, m_{24}, r_{41}, r_{42}, r_{43}, r_{44}, r_5)$ to V .
 4. V checks that $\prod_i (C_K(m_{2i}; r_{4i}) \cdot c_{1i}^{-e}) = c_2$ and $(\prod_{i=1}^4 c_{1i}^{m_{2i}}) \cdot h^{r_5} c^{-e} = c_3$.
-

combined arguments enable one to argue in parallel that $\bigwedge_i y_i = p(\mu_{i1}, \dots, \mu_{in})$ for polynomially many instances of any polynomial p .

As an example, one can construct an argument for the multiplicative relation $\text{AK}(y = C_K(\mu_1, \mu_2, \mu_1\mu_2; \rho))$, $K = (g_1, g_2, g_3; h)$, that is approximately 20% shorter than the argument from [DF02] when using the same security parameters. The argument is based on the idea that $y = C_K(\mu_1, \mu_2, \mu_3; \rho)$ with $\mu_3 = \mu_1\mu_2$ iff A knows such a c_1 that $c_1 = C_{K_1}(\mu_1; \rho_2)$ and $y = C_{K_2}(\mu_1, \mu_2; \rho_3)$, where $K_2 = (g_1, g_2c_1; h)$. (This holds except with a negligible probability.)

The RDF integer tuple commitment scheme exhibits the next *public-key homomorphism property*, the use of which makes many AoK-s more efficient: if $K = (g_1, \dots, g_n; h)$ and $K' = (\prod_i g_i^{a_{1i}} \cdot h^{r_1}, \dots, \prod_i g_i^{a_{ni}} \cdot h^{r_n}; h)$ then

$$C_{K'}(\beta_1, \dots, \beta_n; r) = C_K(\sum_i \beta_i a_{i1}, \dots, \sum_i \beta_i a_{in}; \sum_i \beta_i r_i + r) .$$

B Argument System for Non-negativity

Theorem 4. Let C be the RDF integer tuple commitment scheme, let k be the security parameter and let $\log_2 M = k^{O(1)}$. Let $K = (g; h)$ be the public key. Protocol 1 is a perfectly complete AoK for $\text{AK}(c = C_K(\sum_{i=1}^4 \omega_i^2; \rho))$, or equivalently, for $\text{AK}(c = C_K(\mu) \wedge \mu \geq 0)$. If $\mu \leq M$ then Protocol 1 is HVSZK.

Proof. Proof idea: show that $y = C_K(\sum \nu_i) \wedge \bigwedge (c_i = C_K(\omega_i) \wedge \nu_i = \omega_i^2)$, where all four AoK-s $c_i = C_K(\omega_i) \wedge \nu_i = \omega_i^2$ are done in parallel.

COMPLETENESS. $c^{-e} \cdot \prod_{i=1}^4 C_K(m_{2i}; r_{4i}) = \prod_{i=1}^4 (C_K(m_{1i} + e\omega_i; r_{2i} + er_{1i}) \cdot C_K(-e\omega_i; -er_{1i})) = \prod_{i=1}^4 C_K(m_{1i}; r_{2i}) = c_2$ and $\prod_i c_{1i}^{m_{2i}} \cdot h^{r_5} c^{-e} = \prod_i c_{1i}^{m_{1i}} \cdot \prod_i (C_K(\omega_i; r_{1i}))^{e\omega_i} \cdot h^{r_3 + e \sum_i (1 - \omega_i)r_{1i}} \cdot C_K(-e \sum_i \omega_i^2; -e\rho) = \prod_i c_{1i}^{m_{1i}} \cdot h^{r_3} = c_3$.

HVSZK. The simulator acts as follows. For $i \in [1, 4]$, generate $\tilde{c}_{1i} \leftarrow \mathcal{C}_{Com}$, $\tilde{m}_{2i} \leftarrow \mathbb{Z}_{2^{F(k)M}}$. For $i \in [1, 4]$, generate $\tilde{r}_{4i} \leftarrow \mathbb{Z}_{2^{B+2kF(k)}}$. Generate $\tilde{e} \leftarrow \mathbb{Z}_{F(k)}$, $\tilde{r}_5 \leftarrow \mathbb{Z}_{2^{B+2kF(k)M}}$. Let $\tilde{c}_2 \leftarrow \prod_{i=1}^4 C_K(\tilde{m}_{2i}; \tilde{r}_{4i}) \tilde{c}_{1i}^{-\tilde{e}}$. Let $\tilde{c}_3 \leftarrow \prod_i \tilde{c}_{1i}^{\tilde{m}_{2i}} \cdot h^{\tilde{r}_5} c^{-\tilde{e}}$. The resulting view $((\tilde{c}_{1i})_i, \tilde{c}_2, \tilde{c}_3; \tilde{e}; (\tilde{m}_{2i})_i, (\tilde{r}_{4i})_i, \tilde{r}_5)$ is accepting and has a distribution, statistically close to the distribution of views in a real execution.

To prove that this protocol is specially sound, we must show that from two accepting views, $((c_1)_i, c_2, c_3; e; (m_{2i})_i, (r_{4i})_i, r_5)$ and $((c_1)_i, c_2, c_3; e'; (m'_{2i})_i, (r'_{4i})_i, r'_5)$ with $e \neq e'$, one can efficiently find a tuple $((\omega_i)_i, \rho)$, such that $c = C_K(\sum \omega_i^2; \rho)$. This can be proven as follows. Given such views, $\prod_{i=1}^4 C_K(m_{2i} - m'_{2i}; r_{4i} - r'_{4i}) = \prod_{i=1}^4 c_{1i}^{e-e'}$ and $\prod_i c_{1i}^{(m_{2i}-m'_{2i})} \cdot h^{r_5-r'_5} = c^{e-e'}$. Assuming $K' = (c_{11}, \dots, c_{14}; h)$, this is equivalent to $C_{K'}(m_{21} - m'_{21}, \dots, m_{24} - m'_{24}; r_5 - r'_5) = c^{e-e'}$. By the generalisation of Lemma 1 from [DF02] and by $|e - e'| \in \mathbb{Z}_{F(k)}$, there exists a verifier V^* who together with the arguer A can break the strong divisible root problem with a high probability. \square

Non-interactive version of this argument system is

$$((c_{1i})_i; e \pmod k; (m_{2i}, r_{4i})_{i=1}^4, r_5),$$

where the verifier checks that

$$e \equiv H(c_{11}, \dots, c_{14}, (C_K(m_{2i}; r_{2i})c_{1i}^{-e})_{i=1}^4, c^{-e} \prod_i c_{1i}^{m_{2i}} \cdot h^{r_5}) \pmod{2^k}.$$

The length of non-interactive argument system is $4|C_{Com}| + k + 4(B + 3k + 2 \log_2 F(k) + \frac{1}{2} \log_2 M) + B + 2k + \log_2 F(k) + \frac{1}{2} \log_2 M = 4096 + 80 + 4 \cdot (1024 + 240 + 160) + 1024 + 160 + 80 + \frac{5}{2} \log_2 M = 11136 + \frac{5}{2} \log_2 M$ bits or $1392 + \frac{5}{16} \log_2 M$ bytes.

One can parallelise this argument system even more. Namely, to prove that $y = C_K(\mu; \rho)$, it suffices to prove that $c_i = C_K(\omega_i; r_{1i})$ and $y = (\prod (c_i)^{\omega_i}) (g_i) h^{r_{10}}$, where $r_{10} \leftarrow \rho - r_{11}^2 - \dots - r_{14}^2$.

Verifiable Homomorphic Oblivious Transfer and Private Equality Test

Helger Lipmaa

Laboratory for Theoretical CS

Department of CS&E, Helsinki University of Technology

P.O.Box 5400, FIN-02015 HUT, Espoo, Finland

helger@tcs.hut.fi

Abstract. We describe slightly modified version (that we call the HOT protocol) of the Aiello-Ishai-Reingold oblivious transfer protocol from Eurocrypt 2001. In particular, the HOT protocol will be what we call weakly secure when coupled with many different homomorphic semantically secure public-key cryptosystems. Based on the HOT protocol, we construct an efficient verifiable oblivious transfer protocol and an efficient verifiable private equality test. As a concrete application of our results, we propose a novel protocol called proxy verifiable private equality test, and apply it to a cryptographic auction scheme to improve its security.

Keywords: cryptographic auctions, homomorphic encryption, verifiable oblivious transfer, verifiable private equality test.

1 Introduction

In a two-party $\binom{n}{1}$ -oblivious transfer (OT) protocol the chooser receives a chosen single input from the database of n items, without the sender getting to know which element was retrieved. We first present a concise proof that a slightly modified version (that we call the *homomorphic oblivious transfer* or the HOT protocol) of the $\binom{n}{1}$ -OT protocol of [AIR01] is perfectly sender-private iff for all possible private keys x of the used homomorphic semantically secure public-key cryptosystem, the corresponding plaintext space is a cyclic group of prime order M . Additionally, we show that the HOT protocol is computationally sender-private when M is composite but hard to factor by the chooser. This makes it possible to use the recent Damgård-Jurik cryptosystem [DJ03] in this context.

We then also introduce another security notion for oblivious transfer protocols, *weak sender-privacy*, that is sufficient whenever the oblivious transfer protocol does not have to be chooser-verifiable. Intuitively, a protocol is weakly sender-private if the chooser will never obtain information about more than one item from the database; however, the Chooser can still obtain information about a single item of the database even if his input to the protocol is out of the bounds. We show that the $\binom{n}{1}$ -HOT protocol is weakly sender-private whenever $\mathcal{M}_\Pi(x)$ is a residue class ring with $\Phi(M) > n$, where $\Phi(M)$ is the smallest prime divisor of M . A weakly sender-private $\binom{n}{1}$ -HOT protocol can

be made sender-private by accompanying it with a zero-knowledge argument that chooser's input was in the correct range. In this case, some suitable homomorphic cryptosystems are [El 84, Pai99, DJ01, DJ03], and possibly [NS98, OU98]. Therefore, the $\binom{n}{1}$ -HOT protocol can be based on different hardness assumptions (like the DCRA assumption of Paillier [Pai99]), made to work efficiently with long strings (in the case of Damgård-Jurik cryptosystems [DJ01, DJ03]), and efficiently thresholded (in the case of [El 84, DJ03]).

In a verifiable (also known as “committed” [CvdGT95, CD97, CC00]) oblivious transfer protocol, the chooser obtains sender's commitment to every database element and can later verify if these elements were equal to some other elements, used in other parts of the higher-level protocol. In the new verifiable homomorphic oblivious transfer protocol (Protocol 2), the chooser and the sender execute the HOT protocol so that the chooser obtains the random number that was used by the sender to commit to the chosen database element. Security of the verifiable HOT protocol depends additionally on the security of the employed homomorphic commitment scheme Γ , and on a simple relation between the sizes of plaintext spaces of Π and Γ . In particular, the verifiable HOT protocol based on the ElGamal cryptosystem and on the CGHN commitment scheme [CGHN01] is perfectly sender-private (unlike the recent slightly less efficient verifiable oblivious transfer protocol of [AJL03] that offers only statistical sender-privacy), and allows efficient reconstruction of the transmitted data item (unlike, again, [AJL03]).

After that, we show how to use the ideas, developed while constructing the HOT and the verifiable HOT protocols, in another context. *Private equality test (PET)* [FNW96, NP99, BST01] (let the Chooser to know whether the private inputs W_{Cho} and W_{Sen} of the Chooser and the Sender are equal without leaking any other information) is yet another widely used cryptographic protocol. We propose a new two-round homomorphic PET (HPET) protocol that is very similar to the $\binom{n}{1}$ -HOT protocol. Previously known PET protocols [FNW96, NP99, BST01] were significantly less efficient. The HPET protocol is perfectly sender-private, when based on a homomorphic semantically secure public-key cryptosystem with a prime M like the ElGamal [El 84]. Computational privacy is achieved when the decrypter cannot factor M [DJ03]. As with the HOT protocol, we show how to make the HPET protocol verifiable, although the concrete technique for this will be different.

Finally, we propose a novel application for the new verifiable HPET protocol. Namely, we show that it can be generalised to the proxy verifiable HPET protocol and then use the latter to increase the security of the probably most efficient currently known $((b+1)$ st-price sealed-bid) cryptographic auction scheme without threshold trust by Lipmaa, Asokan and Niemi [LAN02]. More precisely, we show how to make the payment enforcement phase of [LAN02] more secure by not revealing the contract price either to the bidders or to the seller, before all the bidders have shown by using the proxy verifiable HPET protocol whether their bid was equal to the (yet unknown to them) value of the highest bid. We hope to see more applications of the proxy verifiable HPET protocol in

the future, especially since to the best of our knowledge, no efficient proxy PET protocols were known previously at all.

All the proofs in this paper are slightly simplified due to the lack of space.

Road-Map. We start the paper by describing cryptographic building blocks (Section 2). Section 3 defines some properties of the public-key cryptosystems that we need later. Our main contribution starts with Section 4, where we propose the new oblivious transfer protocols and prove their security. In Section 5, we describe a new private equality test protocol, together with some extensions. Finally, in Section 6 we propose some applications of the new protocols. In particular, we demonstrate how to use the proxy verifiable PET protocol in auctions.

2 Preliminaries and Cryptographic Building Blocks

Throughout this paper, let k be the security parameter. We assume that the reader knows standard complexity-theoretic notions like negligibility and probabilistic polynomial time (PPT); we take the latter to be equivalent to “efficiently computable”. For a positive integer x , let $\Phi(x)$ denote the smallest prime divisor of x . Let $\varphi(x)$ be the Euler’s totient function of x . Recall that if $x = \prod_i p_i^{c_i}$ for different primes p_i then $\varphi(x) = x \cdot \prod_i (1 - 1/p_i)$.

For a distribution (random variable) X , let $x \leftarrow X$ denote the assignment of x according to X . We often identify sets with the uniform distributions on them, and algorithms with their output distributions, assuming that the algorithm that outputs this distribution is clear from the context or just straightforward to construct. The statistical difference of two distributions X and Y over the discrete support U is defined as $\Delta(X||Y) := \max_{S \subseteq U} |\Pr[X \in S] - \Pr[Y \in S]|$.

Homomorphic Semantically-Secure Cryptosystems. Let $\Pi = (\mathcal{G}_\Pi, E, D)$ be a public-key cryptosystem, where \mathcal{G}_Π is the key generation algorithm $\mathcal{G}_\Pi : 1^k \mapsto (x, K)$, E is the encryption algorithm $E_K : (m; r) \mapsto E_K(m; r)$ and D is the decryption algorithm $D_K : c \mapsto D_K(c)$. Assume that for every possible private key x , the corresponding message space $\mathcal{M}_\Pi(x)$ is an Abelian group with the group operation $+$, and that the corresponding ciphertext space $\mathcal{C}_\Pi(x)$ is a Abelian group with the group operation \cdot . We denote the space of random coins by $\mathcal{R}_\Pi(x)$. (In particular, this notation indicates that $\mathcal{M}_\Pi(x)$, $\mathcal{R}_\Pi(x)$ and $\mathcal{C}_\Pi(x)$ might be unknown to the encrypter, although this is usually not the case.)

We say that Π is *homomorphic*, if $E_K(m_1; r_1) \cdot E_K(m_2; r_2) = E_K(m_1 + m_2; r_1 \circ r_2)$ for some deterministic binary operation $\circ : \mathcal{R}_\Pi(x)^2 \rightarrow \mathcal{R}_\Pi(x)$. Then $E_K(m; r)^s = E_K(m^s; \text{rf}_e(r, s))$ for another deterministic mapping rf_e . Given that $\text{rf}_e(r, s + 1) = \text{rf}_e(r, s) \circ r$, we will denote $\text{rf}_e(r, s)$ by r^s .

For an algorithm A , define $\text{Adv}_{\Pi, k}^{\text{sem}}(A) := |\Pr[(x, K) \leftarrow \mathcal{G}_\Pi(1^k), (m_0, m_1) \leftarrow A(1^k, K), r \leftarrow \mathcal{R}_\Pi(x), b \leftarrow [0, 1], c \leftarrow E_K(m_b; r) : A(1^k, K, m_0, m_1, c) = b] - \frac{1}{2}|$ to be the advantage that A has over random guessing when trying to distinguish

random encryption of two elements, chosen by herself. We say that Π is *semantically secure* if for all PPT algorithms A , $\text{Adv}_{\Pi,k}^{\text{sem}}(A)$ is negligible in k . This definition is polynomially equivalent to other common definitions of semantical security.

A classical example of an homomorphic semantically secure public-key cryptosystem is the ElGamal public-key cryptosystem [El 84] with $E_K(m; r) = (mh^r; g^r)$; it works over any family of multiplicative groups where the Decisional Diffie-Hellman Assumption is true. In particular, $\mathcal{M}_\Pi(x)$ may be a subgroup of \mathbb{Z}_p^* , generated by an element of order q , where p and q are primes such that $q \mid (p-1)$. In another important case, $\mathcal{M}_\Pi(x)$ is a prime-order subgroup of a suitable elliptic curve group. Another example of an homomorphic semantically secure public-key cryptosystem is the Paillier public-key cryptosystem [Pai99], where as modified by [CGHN01,DJ01], $E_K(m; r) = (1 + mN)r^N \bmod N^2$ for $N = pq$, $\mathcal{M}_\Pi(x) = \mathbb{Z}_N$ and $\mathcal{R}_\Pi(x) = \mathbb{Z}_N^*$. Here, $E_K(m_1; r_1) \cdot E_K(m_2; r_2) = E_K(m_1 + m_2; r_1 r_2)$.

Homomorphic Commitment Schemes. In a commitment scheme $\Gamma = (\mathcal{G}_\Gamma, C)$, the committer sends an element $m \leftarrow \mathcal{M}_\Gamma(x)$ of the plaintext space to the receiver in a committed form, $c \leftarrow C_K(m; r)$, where (x, K) is generated by $\mathcal{G}_\Gamma(1^k)$ and $r \leftarrow \mathcal{R}_\Gamma(x)$. We denote the commitment space of Γ by $\mathcal{C}_\Gamma(x)$. In the context of our paper, all commitment schemes are required to be perfectly (or at least statistically) hiding and computationally binding. More precisely, for an algorithm A , define $\text{Adv}_{\Gamma,k}^{\text{hide}}(A) := |\Pr[(x, K) \leftarrow \mathcal{G}_\Gamma(1^k), (m_0, m_1) \leftarrow A(1^k, K), r \leftarrow \mathcal{R}_\Gamma(x), b \leftarrow [0, 1], c \leftarrow C_K(m_b; r) : A(1^k, K, m_0, m_1, c) = b] - \frac{1}{2}|$ to be the advantage that A has over random guessing when trying to distinguish random commitments of two elements, chosen by herself. We say that Γ is *statistically hiding* if for all (not necessarily PPT) algorithms A , $\text{Adv}_{\Gamma,k}^{\text{hide}}(A)$ is negligible in k . We allow Γ to be a trapdoor commitment scheme. That is, if A has access to the secret key x , she can break the binding property. Γ is *homomorphic* if for any (m_1, m_2, r_1, r_2) , $C_K(m_1; r_1)C_K(m_2; r_2) = C_K(m_1 + m_2; r_1 \circ r_2)$ for some binary operator \circ . We will sometimes assume that $\mathcal{R}_\Gamma(x)$ has a unit element 1.

In the Pedersen commitment scheme [Ped91], the setting is the same as in the ElGamal public-key cryptosystem, and $C_K(m; r) := g^m h^r$ for $r \in \mathcal{R}_\Gamma(x)$. In the CGHN [CGHN01] trapdoor commitment scheme, $N = pq$, $C_K(m; r, s) = (1 + mN)r^N h^s \bmod N^2$, where $h = \alpha^N(1 + \beta N) \bmod N^2$ for random $\alpha \leftarrow \mathbb{Z}_N^*$ and $\beta \leftarrow \mathbb{Z}_N \setminus \{0\}$, $r \leftarrow \mathbb{Z}_N^*$ and $s \leftarrow \mathbb{Z}_N$. Then $C_K(m_1; r_1, s_1)C_K(m_2; r_2, s_2) = C_K(m_1 + m_2; r_1 r_2, s_1 + s_2)$.

$\binom{n}{1}$ -Oblivious Transfer. During an $\binom{n}{1}$ -oblivious transfer protocol, the chooser receives precisely one, chosen by himself, item from the database $\mu = (\mu_1, \dots, \mu_n)$ of n items, maintained by the sender. The sender does not get to know which item was transferred. In the general case, the index i in μ_i does not have to be an integer (indeed, we will not require it in the following), it is sufficient that different elements of μ are indexed by different elements of some set $\mathcal{I} = (\mathcal{I}_1, \dots, \mathcal{I}_n)$. However, for the sake of simplicity we will denote the i th element of the database by μ_i (and not by $\mu_{\mathcal{I}_i}$).

Importantly, most of the cryptography can be based on the oblivious transfer [Kil88]. Additionally, efficient oblivious transfer is necessary, since oblivious transfer is often the most expensive part of cryptographic protocols. An example is Yao's two-party computation model, where the proxy oblivious transfer [NPS99] is the only sub-protocol that requires public-key operations.

The security of an (information-theoretically sender-private) $\binom{n}{1}$ -oblivious transfer protocol is usually defined in two parts. We will follow the definitions of [NP01, Section 2.1.1]. (It is possible to switch the security requirements so as to require information-theoretical chooser-privacy and computational sender-privacy, but corresponding protocols will be out of the scope of this paper. See, e.g., [Tze02].)

Denote a run of interactive protocol between A who has private input a and random tape r_a and between B who has private input b and a random tape r_b as $(A, B)[a, r_a; b, r_b]$. As usually, define Cho 's view $\text{view}_{\text{Cho}}[\sigma, r_{\text{Cho}}; \mu, r_{\text{Sen}}]$ in the oblivious transfer protocol $(\text{Cho}, \text{Sen})[\sigma, r_{\text{Cho}}; \mu, r_{\text{Sen}}]$ as the concatenation of its private input σ , random tape r_{Cho} , the protocol transcript, and its private output μ_σ . The view of Sen is defined dually.

Computational Chooser-Privacy: For an algorithm A executing the sender's part in the oblivious transfer protocol $(\text{Cho}, A)[\sigma, r_{\text{Cho}}; \mu, r_A]$, define $\text{Adv}_{\text{Cho},k}^{\text{otcho}}(A) := \Pr[(\sigma_0, \sigma_1, \mu') \leftarrow A(1^k, \mu, r_A), b \leftarrow [0, 1] : A(1^k, \mu, r_A, \text{view}_A[\sigma_b, r_{\text{Cho}}; \mu', r_A]) = b']$ to be the probability that after observing an execution of the protocol $(\text{Cho}, A)[\sigma_b, r_{\text{Cho}}; \mu, r_{\text{Sen}}]$, A can predict which of the two possible choices σ_0 and σ_1 was used by the chooser. We call an oblivious transfer protocol (*computationally*) *chooser-private* if $\text{Adv}_{\text{Cho},k}^{\text{otcho}}(A)$ is negligible for any PPT algorithm A .

Statistical Sender-Privacy: We make the comparison to the ideal implementation, using a trusted third party that receives μ from the sender, receives σ from the chooser, and tells the chooser μ_σ . We assume that μ_σ is garbage (i.e., a random value from some μ -independent set \mathcal{T}) if $\sigma \notin \mathcal{I}$.

We define the security by showing that for every algorithm A , one can define a simulator S that, given only private input σ , random tape r_A , and private output μ_σ of A , generates output that is statistically indistinguishable from the view of A that reacts with the honest sender Sen . More precisely, for a sender Sen and an algorithm S , define $\text{Adv}_{\text{Sen},k}^{\text{otsen}}(A, S) := \Delta(S(1^k, \sigma, r_A, \mu_\sigma) || \text{view}_A[\sigma, r_A; \mu, r_{\text{Sen}}])$. We say that the oblivious transfer protocol is *statistically sender-private* if for every (not necessarily PPT) A there exists a (not necessarily PPT) S , such that $\text{Adv}_{\text{Sen},k}^{\text{otsen}}(A, S)$ is negligible in k . As usually, sender-privacy is perfect when $\text{Adv}_{\text{Sen},k}^{\text{otsen}}(A, S) = 0$.

As argued, e.g., in [NP01, Section 2.1.2], an oblivious transfer protocol does not have to guarantee the correctness (even if Cho is honest but Sen is not, Cho will still receive Sen 's input μ_σ). Following this convention, also we will leave it up to the application protocols to provide security in this sense.

The next $\binom{n}{1}$ -oblivious transfer (OT) protocol by Aiello, Ishai and Reinhold [AIR01] provides perfect sender-privacy and computational chooser-privacy. Assume that $\Pi = (\mathcal{G}_\Pi, E, D)$ is an homomorphic semantically secure public-key cryptosystem that works over a plaintext space \mathbb{Z}_M of prime order $M =$

$|\mathcal{M}_\Pi(x)|$. The sender **Sen** has a vector $\mu = (\mu_1, \dots, \mu_n) \in \mathbb{Z}_M^n$. The chooser **Cho** has made a choice σ . The AIR protocol works as follows: (a) **Cho** generates a secret/public key pair $(x, K) \leftarrow \mathcal{G}_\Pi(1^k)$. **Cho** generates a random coin $r \leftarrow \mathcal{R}_\Pi(x)$ and computes $c \leftarrow E_K(\sigma; r)$. He sends (K, c) to **Sen**. (b) **Sen** performs the following, for $i \in [1, n]$: Generate random $(r_i, s_i) \leftarrow \mathcal{R}_\Pi(x) \times \mathcal{M}_\Pi(x)$. Compute $c_i \leftarrow E_K(\mu_i; 0) \cdot (c \cdot E_K(-i; 0))^{s_i} \cdot E_K(0; r_i)$. Send c_i to **Cho**. (c) **Cho** obtains $\mu_\sigma \leftarrow D_K(c_\sigma)$. As a consequence, the AIR protocol requires n online encryptions by the sender. A similar but slightly less efficient $\binom{n}{1}$ -OT protocol was independently proposed by Naor and Pinkas [NP01, Section 4.1].

Often one needs a $\binom{n}{1}$ -oblivious transfer protocol to be *sender-verifiable* (also known as “committed”) in the next sense [CvdGT95, CD97, AJL03]: after the oblivious transfer protocol, the chooser obtains sender’s commitment c_i to every database element that can be later used in various zero-knowledge proofs or arguments. Recently, Ambainis, Jakobsson and Lipmaa proposed probably the first two-round verifiable oblivious transfer protocol [AJL03]; their protocol was based on decoupling the Naor-Pinkas oblivious transfer protocol and the Pedersen commitment scheme. Briefly, the Naor-Pinkas protocol uses a sub-protocol to recover a key that was used to encrypt the database element. The Ambainis-Jakobsson-Lipmaa (AJL) protocol uses the same sub-protocol to recover a nonce that was used to commit to the database element.

Private Equality Test. At the end of the private equality test (PET, also known as “comparing information without leaking it” or “socialist millionaires’s problem”) protocol, the Chooser **Cho** gets to know whether Sender’s input W_{Sen} equals to that of the Chooser, W_{Cho} . **Cho** will not get to know anything else about W_{Sen} , while **Sen** should not have any private output at all. Exactly as in the case of oblivious transfer, the security is divided into statistical sender-privacy and computational chooser-privacy. The security definitions are standard and we omit them due to the space constraints.

Previously proposed PET protocols [FNW96, NP99, BST01] had an extra emphasis on developing fair protocols where both the Chooser and the Sender get to know the result of comparison. None of these protocols is however really efficient even when simplified so as not to have the fairness property. For example, the PET protocol from [BST01] requires multiple rounds and zero-knowledge proofs of knowledge. One application, considered at the end of our paper actually relies on the asymmetric nature of our PET protocols.

3 Affine Public-Key Cryptosystems

Next we describe a new property of homomorphic semantically secure public-key cryptosystems that will be necessary in the later described protocols. First, recall that a finite cyclic Abelian group is isomorphic to some residue class group \mathbb{Z}_N . Now, let \mathcal{D} and $\mathcal{D}' \neq 0$ be two distributions of elements of \mathbb{Z} . We say that \mathcal{D}' *affinely ε -approximates* \mathcal{D} on additive group G if for every $g, g' \in G$, $g \neq 0$, $\Delta(\mathcal{D}' \cdot g + g' | \mathcal{D}) \leq \varepsilon$. We call G *ε -affine* if such distributions \mathcal{D} and

\mathcal{D}' exist. We say that G is computationally ε -affine if it is ε -affine under the condition that g and g' must be generated by a PPT algorithm. We say that G is (computationally) non-affine if it is not (computationally) $1/2$ -affine.

Assume that the order of G is public. First, if G is a cyclic group of prime order, one can define $\mathcal{D}' := |G|$ and $\mathcal{D} := G$. Then G is 0-affine. If G is a cyclic group of composite order, $G \cong \mathbb{Z}_M$, then for any generator g of G , all elements ag for $\gcd(a, |G|) = 1$ are generators, while for a with $\gcd(a, |G|) \neq 1$, $|\langle ag \rangle| \leq |G|/2$. Therefore, G is non-affine. On the other hand, if one assumes that it is hard to factor $|G|$ then G will be computationally 0-affine. If G is an acyclic group, then every element $g \in G$ generates a nontrivial subgroup $\langle g \rangle$ of G of order $\leq G/2$. In this case, any choice of $\mathcal{D}' \neq 0$ leads to non-affinity even in the computational sense.

Let $\varepsilon = (\varepsilon_k)$ be a family of probabilities. We say that $\Pi = (\mathcal{G}_\Pi, E, D; \mathcal{S}, \mathcal{T})$ is an ε -affine public-key cryptosystem, if $\Pi' = (\mathcal{G}_\Pi, E, D)$ is a homomorphic semantically secure public-key cryptosystem, \mathcal{S} and \mathcal{T} are PPT algorithms, with $\mathcal{S}(1^k, K) \subset \mathbb{Z}$, $\mathcal{T}(1^k, K) \subseteq \mathcal{M}_\Pi(x)$ with $|\mathcal{T}(1^k, K)| > 1$, and for every security parameter k , key pair $(x, K) \in \mathcal{G}_\Pi(1^k)$, $\text{Adv}_{\Pi, x}^{\text{affine}} := \max_{a \in \mathcal{M}_\Pi(x) \setminus \{0\}, b \in \mathcal{M}_\Pi(x)} \Delta(\mathcal{S}(1^k, K)a + b | \mathcal{T}(1^k, K)) \leq \varepsilon_k$. Therefore, Π is *perfectly affine* if for every x , $\mathcal{M}_\Pi(x)$ is a cyclic group with known prime order. We say that Π is *computationally affine* if for every x , $\mathcal{M}_\Pi(x)$ is a cyclic group with known composite order under the assumption that it is hard even for the decrypter to factor M . (If M is not known, perfect affinity may change to statistical affinity.)

4 Homomorphic Oblivious Transfer Protocols

Simplified notation. To simplify the notation, from now on we will omit the arguments $(1^k, K)$ of \mathcal{S} and \mathcal{T} , the argument x of \mathcal{M}_Π and \mathcal{R}_Π , and the argument \tilde{x} of \mathcal{M}_Γ and \mathcal{R}_Γ .

4.1 Simpler Protocol without Sender-Verifiability

Protocol 1 depicts the new homomorphic oblivious transfer protocol. A very similar protocol was proposed in [AIR01]; we will provide comparisons later in this section.

Theorem 1. *Let k be the security parameter. Let $\Pi = (\mathcal{G}_\Pi, E, D; \mathcal{S}, \mathcal{T})$ be a (statistically or computationally) ε -affine homomorphic semantically secure public-key cryptosystem for some $\varepsilon = (\varepsilon_k)_k$. Let the database size n be polynomial in k . The HOT protocol depicted by Protocol 1 is a secure oblivious transfer protocol between the chooser **Cho** and the sender **Sen** in the next sense. When Π is semantically secure, then the HOT is computationally chooser-private. Let $M = |\mathcal{M}_\Pi|$. Sender's privacy is (a) perfect when $\varepsilon_k = 0$, (b) computational, with the best adversary having success $n\varepsilon_k$ when ε_k is negligible in k and Π is computationally ε -affine.*

Protocol 1 The homomorphic oblivious transfer protocol.PRIVATE INPUT: Cho has an index $\sigma \in \mathcal{I}$, Sen has $\mu = (\mu_1, \dots, \mu_n)$.PRIVATE OUTPUT: Cho has μ_σ .

1. The chooser generates a new key pair $(x, K) \leftarrow \mathcal{G}_\Pi(1^k)$, a random coin $r \leftarrow \mathcal{R}_\Pi$, and sets $c \leftarrow E_K(\mathcal{I}_\sigma; r)$. He sends (K, c) to the sender.
2. For $i \in [1, n]$, the sender chooses random $s_i \leftarrow \mathcal{S}$ and $r_i \leftarrow \mathcal{R}_\Pi$, computes $c_i \leftarrow E_K(\mu_i; 0) \cdot (c \cdot E_K(-\mathcal{I}_i; 0))^{s_i} \cdot E_K(0; r_i)$, and sends c_i to the chooser.
3. The chooser outputs $\mu_\sigma \leftarrow D_K(c_\sigma)$.

Proof. CORRECTNESS: If both players are honest then $c_i = E_K(\mu_i + s_i(\sigma - i); r^{s_i} \circ r')$ and $D_K(c_\sigma) = \mu_\sigma$, and thus this protocol is correct.

CHOOSER-PRIVACY: If the sender can distinguish the views $\{E_K(\sigma; \mathcal{R}_\Pi)\}$ and $\{E_K(\sigma'; \mathcal{R}_\Pi)\}$ then Π is not semantically secure. (More precisely, if one can violate the chooser-privacy in time t with probability δ , then one can violate the semantical security of Π in time $t + \text{const}$ and with probability δ .)

STATISTICAL SENDER-PRIVACY: We construct the next unbounded simulator S of A : S executes A instruction-by-instruction, except that when A sends a message c to the sender Sen, S interrupts and answers to c with (c_1, \dots, c_n) , where c_i is computed as follows: if $i := D_K(c) \in \mathcal{I}$ then $c_i \leftarrow c^{s_i} \cdot E_K(\mu_i - s_i D_K(c); \mathcal{R}_\Pi)$ for random $s_i \leftarrow \mathcal{S}$, otherwise $c_i \leftarrow E_K(\mathcal{T}; \mathcal{R}_\Pi)$.

Now, if $\sigma := D_K(c) \notin \mathcal{I}$ (the opposite case $D_K(c) \in \mathcal{I}$ is analogous), the output distribution of the simulator (for fixed random tape ρ of S , and for fixed c) is $(\rho; c; \dots, E_K(\mathcal{T}; \mathcal{R}_\Pi), \dots; \mu_\sigma)$, while the output distribution of A is $(\rho; c; \dots, c^{s_i} \cdot E_K(\mu_i - s_i \mathcal{I}_\sigma; \mathcal{R}_\Pi), \dots; \mu_\sigma)$ for random $s_i \leftarrow \mathcal{S}$. For a fixed c , the difference between these two distributions is

$$\text{Adv}_{\text{Sen}, k}^{\text{otsen}}(A, S) \leq n \cdot \max_{a \neq 0} \Delta(E_K(Sa + \mu_i; \mathcal{R}_\Pi) || E_K(\mathcal{T}; \mathcal{R}_\Pi)) \leq$$

$$n \cdot \max_{a \neq 0, b} \Delta(E_K(Sa + b; \mathcal{R}_\Pi) || E_K(\mathcal{T}; \mathcal{R}_\Pi)) = n \cdot \max_{a \neq 0, b} \Delta(Sa + b || \mathcal{T}) = n \cdot \text{Adv}_{\Pi, x}^{\text{affine}}.$$

Both claims follow straightforwardly. \square

Weak Server-Privacy. Only a few homomorphic semantically secure public-key cryptosystems are affine, as seen from Table 1. Fortunately, it comes out that the HOT protocol is sender-private under much broader settings when we slightly weaken the security definitions.

We say that the oblivious transfer protocol provides *weak sender-privacy* if the chooser will retrieve more than an ideal amount of information about at most one value μ_i , where $i = \sigma$ when the Chooser has private input $\sigma \in \mathcal{I}$. Weak sender-privacy is sufficient in almost all cases when the oblivious transfer protocol is not required to be chooser-verifiable. (Chooser-verifiability can be defined as the requirement that the chooser must be able to prove that the database element she received was indexed by her choice.) An example application where weak sender-privacy is sufficient is the paid database queries setting,

where the database maintainer is only interested in the number of the items that the client will obtain, and not that the indices of the obtained items satisfy any requirements.

Often (as in the case of the oblivious transfer protocol, proposed in Sect. 4), a weakly sender-private oblivious transfer protocol can be transferred to a statistically sender-private one by accompanying it with a suitable zero-knowledge proof (or argument) that $\sigma \in \mathcal{I}$. Importantly, as we will see from the next theorem, there exist settings where the new oblivious transfer protocol is weakly sender-private but not statistically sender-private.

Theorem 2. *Assume the same setting as in Theorem 1. Additionally, assume that \mathcal{M}_Π is a cyclic group with a generator g , $\mathcal{I}_i = ig$ and that $\Phi(M) > n$. Then the HOT protocol is weakly sender-private. Moreover, a statistically weakly sender-private HOT protocol can be made statistically sender-private if before the second step of Protocol 1, the chooser argues in statistical zero-knowledge that c is an encryption of σ for some $\sigma \in \mathcal{I}$.*

Proof (Sketch). As in Theorem 1, the advantage $\text{Adv}_{\text{Sen},k}^{\text{otsen}}(A, S)$ is bound by $n \max_{a \neq 0, b} \Delta(\mathcal{S}ag + b || \mathcal{T})$. Define $\mathcal{S} := \mathbb{Z}_M$ and $\mathcal{T} := \mathcal{M}_\Pi$. When $a = (\pm\sigma)g$ for $\text{gcd}(\sigma, M) = 1$ then $\mathcal{S}a + b = \mathcal{T}$ for any b . If $a = \sigma g$ for $\text{gcd}(\sigma, M) \neq 1$ then the chooser will see $n - 1$ random encryptions that are distributed as $E_K(\mathcal{T}; \mathcal{R}_\Pi)$, and one encryption of a value $E_K(\mu_i + \mathcal{S}(\sigma - i)g; \mathcal{R}_\Pi)$, this is since $\Phi(M) \geq n$. From the latter she might be able to derive some information about μ_i but this is allowed by the security definition.

The second claim of the theorem (about the zero-knowledge argument) is straightforward. Moreover, if \mathcal{I}_i is encoded as g^i for some group element $g \in \mathcal{M}_\Pi$ then one can show efficiently that $j \in \mathcal{I}$ by using protocols from [DJ01, LAN02]; for $\mathcal{I}_i = i$ the corresponding proofs can be found from [Bou00, Lip03]. (See [Lip03] for some other possible encodings.) \square

Comparison with [AIR01]. The HOT protocol is a generalisation of the protocol of Aiello, Ishai and Reingold [AIR01, Section 5] to a wider selection of plaintext spaces. (Namely, [AIR01] considered only the case when M is a prime.) Careful specification of parameters and the definition of affine cryptosystems allowed us to prove that the protocol is “almost” as secure in cases, not considered in [AIR01]. In particular, as argued earlier, weak sender-privacy is sufficient always when one does not require chooser-verifiability. In most of the real-life scenarios, one does not require chooser-verifiability; in almost all such cases, one can use weakly sender-private variants of the HOT protocol that were not considered in [AIR01]. However, when chooser-verifiability is needed, one will also usually need sender-verifiability, a property not provided by HOT protocol and thus also not by the AIR protocol from [AIR01]. (See Section 4.2 for a new sender-verifiable oblivious transfer protocol.)

Discussion. Importantly, one has quite a flexible choice between possible underlying homomorphic semantically secure public-key cryptosystem Π when

Table 1. Some homomorphic semantically secure public-key cryptosystems Π that make the HOT protocol at least weakly sender-private. The middle column shows whether the corresponding PET protocol from Section 5 is secure.

Π	Sender-privacy	Weak sender-privacy
Sender-private HOT		
[El 84]	Yes (perfect)	Yes (perfect)
[DJ03]	Yes (computational)	Yes (perfect)
Weakly sender-private HOT		
[Pai99]	No	Yes (perfect)
DJ01 [DJ01]	No	Yes (perfect)
[NS98]	No	If $\Phi(M)$ is large (perfect)
[OU98]	No	If $\Phi(p-1)$ is large (statistical)

one only goes for the weak sender-security. Table 1 shows that the HOT is weakly sender-private based on most of the widely known homomorphic semantically secure public-key cryptosystems, and statistically sender-private when based on two known homomorphic semantically secure public-key cryptosystems. From the mentioned homomorphic semantically secure public-key cryptosystems, [NS98] offers a flexible choice of the value $\Phi(M)$ in the range $[3, 2^{11}]$, and for other public-key cryptosystems, $\Phi(M)$ is anyways required to be large for the public-key cryptosystem to be semantically secure. (However, it is not known whether the Naccache-Stern cryptosystem is semantically secure if M is known to Sen.) The Okamoto-Uchiyama public-key cryptosystem [OU98] is a notable exception since there M is not public, and $\Phi(M)$ is not required to be large. Still, even in this case one gets statistical weak sender-privacy by choosing $\mathcal{S} = \mathbb{Z}_{2^{k+\ell/2}}$, where ℓ is the key length.

If combined with the Damgård-Jurik cryptosystem from [DJ03], it becomes possible to use extremely large message spaces. If combined with the ElGamal cryptosystem, one can easily distribute the role of the sender. From the strictly efficiency point of view, the best underlying homomorphic semantically secure public-key cryptosystem would be the ElGamal based on (say) elliptic curves and \mathcal{I}_i is defined as g^i for some generator g . Then $c \leftarrow (g^\sigma h^r, g^r)$ and $c_i \leftarrow (\mu_i g^{(\sigma-i)s_i} h^{rs_i+r_i}; g^{rs_i+r_i})$.

4.2 Verifiable HOT Protocol

Protocol 1 by itself is not (sender-)verifiable but it can be made verifiable by borrowing some ideas from the recent AJL verifiable oblivious transfer protocol by Ambainis, Jakobsson and Lipmaa [AJL03]. More precise, we use the HOT protocol so that the chooser obtains a random nonce m_σ that is used also when the sender commits to μ_σ . The chooser will thus only be able to recover the value of μ_σ . On the other hand, for every i , the sender commits to μ_i , using a random value $\text{tr}(m_i)$ that is known to her. This means that she can use standard zero-knowledge techniques to prove properties of μ_i even for $i \neq \sigma$.

Protocol 2 The verifiable HOT protocol.PRIVATE INPUTS: Cho has σ , Sen has μ .PRIVATE OUTPUTS: Cho obtains μ_σ .

1. Cho creates a key pair $(\tilde{x}, \tilde{K}) \leftarrow \mathcal{G}_\Gamma(1^k)$ and a key pair $(x, K) \leftarrow \mathcal{G}_\Pi(1^k)$. Cho creates a random $r \leftarrow \mathcal{R}$ and computes $c \leftarrow E_K(\mathcal{I}_\sigma; r)$. He sends (K, \tilde{K}, c) to Sen.
2. For all i , Sen creates random $r_i \leftarrow \mathcal{R}$ and $(m_i, s_i) \leftarrow \mathcal{T} \times \mathcal{S}$, computes $v_i \leftarrow (c \cdot E_K(-\mathcal{I}_i; 0))^{s_i} \cdot E_K(m_i; r_i)$ and $c_i \leftarrow C_{\tilde{K}}(\mu_i; \text{tr}(m_i))$. She sends (v_i, c_i) to Cho.
3. Cho outputs $\tilde{\mu}_\sigma \leftarrow \text{retrieve}(c_\sigma \cdot C_{\tilde{K}}(0; \text{tr}(D_K(v_\sigma))^{-1}))$.

Theorem 3. *Let k be the security parameter. Assume that $\Pi = (\mathcal{G}_\Pi, E, D; \mathcal{S}, \mathcal{T})$ is an ε -affine homomorphic semantically secure public-key cryptosystem and that Γ is a homomorphic perfectly hiding commitment scheme. For fixed $(x, K) \leftarrow \mathcal{G}_\Pi(1^k)$ and $(\tilde{x}, \tilde{K}) \leftarrow \mathcal{G}_\Gamma(1^k)$, assume the existence of two deterministic PPT functions $\text{tr} : \mathcal{M}_\Pi \rightarrow \mathcal{R}_\Gamma$ and $\text{retrieve} : C_{\tilde{K}}(m; 1) \mapsto m$. Then Protocol 2 is (a) perfectly sender-private if Γ is perfectly hiding, tr is an injection, $|\mathcal{M}_\Pi| = |\mathcal{R}_\Gamma|$ is a prime and \mathcal{T} and \mathcal{S} are defined as usually; (b) statistically sender-private if Γ is statistically hiding, $(|\mathcal{M}_\Pi| - |\mathcal{R}_\Gamma|)/|\mathcal{R}_\Gamma|$ is negligible and tr is a suitable mapping.*

Proof. CORRECTNESS: If parties are honest then $v_i = E_K(s_i(\mathcal{I}_\sigma - \mathcal{I}_i) + m_i; s_i r + r_i)$, $c_i = C_{\tilde{K}}(\mu_i; \text{tr}(m_i))$ and thus $D_K(v_\sigma) = m_\sigma$, $\tilde{\mu}_\sigma = \text{retrieve}(c_\sigma \cdot C_{\tilde{K}}(0; \text{tr}(m_\sigma) \cdot \text{tr}(m_\sigma)^{-1})) = \text{retrieve}(C_{\tilde{K}}(\mu_\sigma; 1)) = \mu_\sigma$.

CHOOSER-PRIVACY: straightforward, given that Π is semantically secure.

SENDER-PRIVACY: Assume $D_K(c) = \mathcal{I}_\sigma$ for $\sigma \in [1, n]$ (the opposite case is analogous). Denote the distribution $C_{\tilde{K}}(\mathcal{M}_\Gamma; \mathcal{R}_\Gamma)$ by Z and the distribution $((E_K(\tilde{m} + \mathcal{S}(\mathcal{I}_\sigma - \mathcal{I}_i); \mathcal{R}_\Pi), C_{\tilde{K}}(\mu_i; \text{tr}(\tilde{m})))$, where $\tilde{m} \leftarrow \mathcal{T}$, by Y_i . We construct the next unbounded simulator S for A : S executes A step-by-step, except that when A makes a query c to the sender Sen, S interrupts and answers it with $(v_1, c_1, \dots, v_n, c_n)$, where (v_i, c_i) is computed as follows: $(v_i, c_i) \leftarrow (E_K(\mathcal{T}; \mathcal{R}_\Pi), Z)$ when $i \neq \sigma$, and $(v_i, c_i) \leftarrow Y_\sigma$ when $i = \sigma$.

Then the advantage of A is

$$\begin{aligned}
\text{Adv}_{\text{Sen}}^{\text{otsen}}(k)(A, S) &\leq \sum_{i \neq \sigma} \Delta(Y_i || (E_K(\mathcal{T}; \mathcal{R}_\Pi), Z)) \\
&\leq \sum_{i \neq \sigma} \max_{a \neq 0, b} \Delta((Sa + b, C_{\tilde{K}}(\mu_i; \text{tr}(\mathcal{T}))) || (\mathcal{T}, Z)) \\
&\leq \sum_{i \neq \sigma} \max_{a \neq 0, b} \Delta(Sa + b || \mathcal{T}) + \sum_{i \neq \sigma} \Delta(C_{\tilde{K}}(\mu_i; \text{tr}(\mathcal{T})) || Z) \\
&\leq n \cdot \left(\text{Adv}_{\Pi, x}^{\text{affine}} + \Delta(\text{tr}(\mathcal{T}) || \mathcal{R}_\Gamma) + \text{Adv}_{\Gamma, k}^{\text{hide}}(A) \right).
\end{aligned}$$

The claim follows. \square

Straightforwardly, for the weak sender-privacy it suffices to replace the requirement that $\text{Adv}_{\Pi, x}^{\text{affine}}$ is negligible in k by the requirement that $\Phi(\mathcal{M}_\Pi) > n$.

Table 2. Comparison of some verifiable oblivious transfer protocols, with specified homomorphic semantically secure public-key cryptosystem Π and homomorphic commitment scheme Γ . Here we have always $\mathcal{T} = \mathcal{S} = \mathbb{Z}_{|\mathcal{R}_\Gamma|}$ and thus $\text{tr}(m) = m$.

Π	Γ	Sender's priv.	retrieve(c)	Verifiable	Online work (exp/enc/comm)
Naor-Pinkas [NP01]					
ElGamal	(Pedersen)	Perfect	Easy (decryption)	No	$4n/n/-$
AIR [AIR01] and HOT (this paper)					
ElGamal	—	Perfect	Easy (decryption)	No	$-/n/-$
Ambainis-Jakobsson-Lipmaa [AJL03]					
ElGamal	Pedersen	Statistical	Hard (DL)	Yes	$4n/n/n$
Verifiable HOT (this paper)					
ElGamal	Pedersen	Perfect	Hard (DL)	Yes	$-/n/n$
ElGamal	CGHN	Statistical	$(c-1)/N \bmod N^2$	Yes	$-/2n/n$

Comparison with Previous Work. Recall that the up to now most efficient (and the only two-round) verifiable oblivious transfer protocol by Ambainis-Jakobsson-Lipmaa protocol [AJL03] was statistically private, and at the end of the AJL protocol, the chooser had to compute discrete logarithm to recover the value of μ_σ . The verifiable HOT protocol from Protocol 2 solves either—but not both—of these problems, when based on suitable Π and Γ . See Table 2 for a comparison of the verifiable HOT protocol (with the ElGamal cryptosystem but different Γ) with some previous work.

When Γ is the Pedersen commitment scheme with $x = \tilde{x}$ and $K = \tilde{K}$, and $\mathcal{I}_i = g^i$ for some generator g , the resulting scheme will be somewhat similar to [AJL03] with $v_i = (g^{s_i(\sigma-i)} m_i h^{s_i r + r_i}, g^{s_i r + r_i})$. Then $\mathcal{R}_\Gamma = \mathcal{M}_\Pi = \mathbb{Z}_q$, tr is the identity function, $\mathcal{S} = \mathcal{T} = \mathbb{Z}_q$, and the resulting protocol will be both computationally chooser-private and perfectly sender-private under the DDH assumption. (Recall that the AJL protocol from [AJL03] was only statistically sender-private.) Similarly to [AJL03], the drawback of this protocol is that the chooser obtains $C_{\tilde{K}}(\mu_\sigma; 0) = g^{\mu_\sigma}$, from which he has to recover μ_σ by computing a discrete logarithm.

The use of the CGHN [CGHN01] trapdoor commitment scheme as Γ enables one to get rid of the latter drawback with the cost of making the protocol only statistically sender-private. Recall that in the CGHN commitment scheme the chooser recovers $\tilde{c}_\sigma = C_{\tilde{K}}(\mu_\sigma; 1) = (1 + \mu_\sigma N) \bmod N^2$, from which he can efficiently compute $\mu_\sigma = (\tilde{c}_\sigma - 1)/N \bmod N^2$. However, in this case $|\mathcal{R}_\Gamma| \approx |\mathcal{M}_\Pi|^2$, assuming that the public keys of Π and Γ have the same length. There are at least three different methods for overcoming this obstacle: (a) Choosing twice longer keys for the public-key cryptosystem, so that $|\mathcal{M}_\Pi| \geq |\mathcal{R}_\Gamma| \approx N^2$; this might however be impractical; (b) Setting tr to be a pseudorandom number generator; this results in a mere computational privacy; (c) Letting Sen to generate two different random numbers m_i and m'_i , and to use the HOT protocol twice so that the Chooser obtains both m_i and m'_i , and then use both to commit

Protocol 3 New PET protocol, where $\Pi = (\mathcal{G}_\Pi, E, D; \mathcal{S}, \mathcal{T})$ is an affine homomorphic semantically secure public-key cryptosystem.

PRIVATE INPUTS: Chooser has W_{Cho} , Sender has W_{Sen} .

PRIVATE OUTPUTS: Chooser has 0 if $W_{\text{Cho}} = W_{\text{Sen}}$ or garbage, otherwise.

1. Chooser generates a new key pair $(x, K) \leftarrow \mathcal{G}_\Pi(1^k)$, a random $r \leftarrow \mathcal{R}_\Pi$, and sets $c \leftarrow E_K(W_{\text{Cho}}g; r)$. He sends (K, c) to Sender.
 2. Sender generates random $s \leftarrow \mathcal{S}$ and $r' \leftarrow \mathcal{R}_\Pi$. She sends $c' \leftarrow (c \cdot E_K(-W_{\text{Sen}}g; 0))^s \cdot E_K(0; r')$ to the Chooser.
 3. Chooser accepts that $W_{\text{Cho}} = W_{\text{Sen}}$ iff $D_K(c') = 0$.
-

to μ_i . In all three cases, $\text{Adv}_{\text{Sen}}^{\text{otsen}}(k)(A, S) \leq 2n\Delta(\tau(\mathcal{T})||\mathcal{R}_\Gamma)$ is negligible. We suggest, even if this results in a slightly less efficient protocol, to use the third recommendation.

5 Private Equality Test and Enhancements

The Homomorphic Private Equality Test Protocol. Assume that a possible wealths W is encoded as Wg for a generator g of the cyclic group \mathcal{M}_Π . (Other encodings might also work) The new homomorphic private equality test (HPET) protocol (Protocol 3) is in a sense just a—although not a straightforward—simplification of the HOT protocol. Namely, it corresponds to the conditional disclosure of a single element $\mu_{W_{\text{Sen}}} = 0$, where instead of $i = W_{\text{Sen}}$, the sender uses $i = W_{\text{Cho}}$. Thus, $\mu_{W_{\text{Sen}}} = 0$ will be revealed only when $W_{\text{Sen}} = W_{\text{Cho}}$; otherwise the chooser will obtain a random element of \mathcal{M}_Π . Therefore, unsurprisingly, the PET protocol is sender-private exactly when based on a Π that also makes the HOT protocol sender-private.

Theorem 4. *Let k be the security parameter. Assume that $\Pi = (\mathcal{G}_\Pi, E, D; \mathcal{S}, \mathcal{T})$ is an ε -affine homomorphic semantically secure public-key cryptosystem, such that it is computationally hard for the decrypter to factor $M \leftarrow |\mathcal{M}_\Pi|$ for any $x \leftarrow \mathcal{G}_\Pi(1^k)$.*

Let $W_{\text{Sen}} \in \mathcal{M}_\Pi$ and $W_{\text{Cho}} \in \mathcal{M}_\Pi$ be Sender's and Chooser's inputs. Let \mathcal{M}_Π be a cyclic group with generator g . Then Protocol 3, denoted as HPET, is chooser-private. Moreover, (a) if \mathcal{M}_Π is a cyclic group of public prime order, then the HPET protocol is perfectly correct and sender-private, and (b) if \mathcal{M}_Π is a cyclic group of public composite order, where it is hard for the chooser and the sender to factor $|\mathcal{M}_\Pi|$, then this protocol is computationally correct and sender-private.

Proof. CORRECTNESS: When both parties are honest then $c' = E_K(s(W_{\text{Cho}} - W_{\text{Sen}})g; r^s \circ r')$. Thus, $m = 0$ iff (a) $W_{\text{Sen}} = W_{\text{Cho}}$ or (b) $M \mid s(W_{\text{Cho}} - W_{\text{Sen}})g$. The latter can only happen when $\gcd(s(W_{\text{Cho}} - W_{\text{Sen}}), M) \neq 1$, that is, when M is composite, and either the chooser or the sender can find factors of M . (As previously, we will not care about correctness in the case when Sender is

dishonest, leaving it up to an higher level protocol to deal with that.) CHOOSER-PRIVACY: follows straightforwardly from the semantical security.

STATISTICAL SENDER-PRIVACY (Sketch): In this case, the simulator S knows an answer to the question $W_{\text{Sen}} \stackrel{?}{=} W_{\text{Cho}}$ and nothing more about the Sender's wealth. He answers the query c with c' , distributed as $E_K(\mathcal{T}; \mathcal{R}_\Pi)$, if $D_K(c) \neq W_{\text{Sen}}$, and as $E_K(0; \mathcal{R}_\Pi)$ if $D_K(c) = W_{\text{Sen}}$. Clearly, the difference between S 's output and the real view is $\leq \Delta(E_K(\mathcal{S}(W_{\text{Cho}} - W_{\text{Sen}})g; \mathcal{R}_\Pi) || E_K(\mathcal{T}; \mathcal{R}_\Pi)) \leq \text{Adv}_{\Pi, x}^{\text{affine}}$. \square

The HPET protocol is severely more efficient than the BST (Boudot-Schoenmakers-Traoré) protocol [BST01] or the protocol from [NP99]. However, the later can be modified (with significant cost in efficiency) so as to provide fairness, i.e., to guarantee that the Sender will only get to know whether $W_{\text{Sen}} = W_{\text{Cho}}$ if also the Chooser will get to know that. It is unclear yet if our protocol can be modified to become fair, but this is also not our intention.

Unfortunately, the number of currently known homomorphic cryptosystems where the decryption can be performed without knowing the factorisation of $|\mathcal{M}_\Pi|$ is small: the only known examples are [El 84, DJ03]. (See the second column of Tbl. 1.)

Verifiable PET. (Sketch.) Here, we use the same notation as in previous theorems. In a verifiable PET protocol, the Chooser sends $c \leftarrow E_K(W_{\text{Cho}}; r)$ to the Sender, who replies with (v, c') , where $v \leftarrow E_K(s(W_{\text{Cho}} - W_{\text{Sen}})g + m; r^s \circ r')$ and $c' \leftarrow C_{\tilde{K}}(W_{\text{Sen}} \cdot \tilde{g}; \text{tr}(m))$, for $m \leftarrow \mathcal{T}$. Here, $\text{tr} : \mathcal{M}_\Pi \rightarrow \mathcal{R}_\Gamma$ and \tilde{g} is an element of \mathcal{M}_Γ of order at least \mathcal{M}_Π . Clearly, this protocol is correct and secure under reasonable assumptions. The security proof is similar to that, presented in Theorem 3.

Proxy Verifiable HPET. In the $\binom{n}{1}$ -proxy private equality test there is one Alice, n different “Bobs” B_1, \dots, B_n , and a new party called Peggy the Proxy. At the end of the proxy PET protocol, Peggy will get to know whether Alice is as wealthy as B_i , Bob the i th, for all $i \in [1, n]$, while neither Alice nor any of B_1, \dots, B_n will obtain any new information information. Next, we propose a proxy verifiable homomorphic private equality test protocol (see Protocol 4) that bases on a ε -affine homomorphic semantically secure public-key cryptosystem $\Pi = (\mathcal{G}_\Pi, E, D; \mathcal{S}, \mathcal{T})$ that satisfies the same requirements as Π in Thm. 4. (We omit the security proofs.)

This protocol is basically a modification of the HPET protocol with a proxy Peggy who transmits Alice's and B_i 's messages to their partners. As a drawback, Protocol 4 reveals W_A to Peggy on step 5, but importantly, this only happens after Peggy has committed to B_i 's answers: if Peggy would get to know x before forwarding (K, \tilde{K}, c) on step 2, she might be able, in collaboration with some B_i , to stop the protocol before sending the commitment χ to Alice if the outcome is not suitable for Peggy. This attack is relevant in, e.g., the auction scenario (see Sect. 6), and is one of the reasons why x is sent to Peggy only at the end of the

Protocol 4 The proxy verifiable HPET protocol.

PRIVATE INPUTS: Alice has W_A , B_i has W_{B_i} . PRIVATE OUTPUTS: For all i , Peggy has 0 if $W_A = W_{B_i}$ or garbage, otherwise.

1. Alice generates new private key pairs $(x, K) \leftarrow \mathcal{G}_\Pi(1^k)$ and $(\tilde{x}, \tilde{K}) \leftarrow \mathcal{G}_\Gamma(1^k)$, a random $r \leftarrow \mathcal{R}_\Pi$, and sets $c \leftarrow E_K(W_A; r)$. She sends (K, \tilde{K}, c) to Peggy.
2. Peggy forwards (K, \tilde{K}, c) to players B_1, \dots, B_B .
3. For every i , B_i creates a random $m_i \leftarrow \mathcal{T}$, computes $v_i = E_K(m_i + s_i(W_A - W_{B_i}); r^{s_i} \circ r'_i)$ for random $s_i \leftarrow \mathcal{S}$ and $r'_i \leftarrow \mathcal{R}_\Pi$, and sets $c_i \leftarrow C_{\tilde{K}}(W_{B_i}; \text{tr}(m_i))$. He sends (v_i, c_i) together with his signature over (K, \tilde{K}, c, c, v) to Peggy.
4. Peggy collects all values $\{v_i, c_i\}$, and signs (at an a priori fixed time) their joint commitment. He sends the signed commitment χ to Alice.
5. Alice sends W_A, x and her signature on (W_A, χ, x) to Peggy.
6. For every i , Peggy decrypts v_i by using the key x , and obtains a message $\tilde{m}_i \in \mathcal{M}_\Pi$. She decides that $W_A = W_{B_i}$ iff $c_i = C_{\tilde{K}}(W_A; \text{tr}(\tilde{m}_i))$.

protocol. As we will also see in Sect. 6, in some applications revealing W_A at the end of the protocol is actually desirable.

Second, More Secure, Proxy Verifiable HPET Protocol. (Sketch.) In an alternative protocol to Protocol 4, instead of sending x to Peggy, Alice receives (v, c) from Peggy, obtains all messages \tilde{m}_i , and then proves in zero-knowledge whether v_i commits to W_A for all $i \in [1, n]$. This protocol is obviously more secure than the first protocol (since x and thus also W_A will not be revealed to Peggy), but requires at least one additional round and more communication.

6 Applications

Applications of the Verifiable Oblivious Transfer Protocol. In [AJL03], Ambainis, Jakobsson and Lipmaa proposed several protocols for the cryptographic randomised response technique. Their first protocol—that bases on their own verifiable oblivious transfer protocol—can be made more efficient (and also perfectly private for the respondent) by using the verifiable HOT protocol instead. Note that at least in their application a weakly sender-private oblivious transfer protocol with a trapdoor commitment scheme will be sufficient. See, e.g., [CvdGT95, CD97, CC00] for more applications for the verifiable HOT protocol.

Auctions. The LAN auction scheme [LAN02] is (probably) the most efficient secure cryptographic $(b+1)$ st auction scheme without threshold trust; in large-scale auctions with many participants it requires 10–100 times less communication than the Naor-Pinkas-Sumner scheme [NPS99]. On the other hand, the LAN scheme has two principal drawbacks. First, the involved trusted auction authority A will get to know the bid statistics. As argued in [LAN02], this is

not a weakness from the economic viewpoint when relying on the assumption that the occasional seller and the well-established business authority A do not collaborate.

Second, the LAN scheme has only an optimistic payment enforcement procedure. Namely, after the seller has received the value of the b th highest bid X_b from A , reliable winner determination is only possible when all the bidders (or at least b highest bidders) will complete a zero-knowledge proof that shows whether they bid more than X_b or not. Clearly, it may be difficult to force the bidders to collaborate at this time—especially after they know the value of X_b —, and it may be hard to distinguish between the malicious bidders (who want to disrupt the auctions, lose their interest in participation since they are not winning, or are not willing to pay as much), shills and bidders that have some genuine problems with their software or hardware. Moreover, some bidders might object to such enforcement even if they have no desire to cheat, by whatever moral or psychological reasons.

By using the proxy verifiable HPET protocol (Protocol 4), one can eliminate the second problem of the LAN scheme for $b \leq 1$ with a moderate increase in the communication complexity. The basic idea of our solution is that after the third party A has computed the b th highest bid X_b , he will not send X_b to the seller P , as it was done in the original protocol of [LAN02]. Instead, the seller will act as a proxy in $(b - 1)$ parallel proxy verifiable HPET protocols with the inputs X_1, \dots, X_{b-1} from A and the input b_i (B_i 's bid) from the bidder B_i . After the 3rd step of the proxy verifiable PET protocol, neither the seller nor any of the bidders knows X_j for any j . Thus, none of the bidders (including the shills who cooperate with the auctioneer) has the motivation to discontinue participation in the auction. In particular, the seller has no better strategy than to be honest in step 4 of Protocol 4. Moreover, he will receive X_1, \dots, X_{b-1} only on step 5 of the proxy verifiable HPET protocol, after his commitment and thus his actions are accountable. The drawback of this solution is that the seller will get to know X_1, \dots, X_{b-1} .

Alternatively, the participants can use the alternative proxy verifiable HPET protocol that was sketched before; in this case, no X_j will be leaked to the seller, but the communication complexity of the whole scheme increases somewhat, since the authority must provide $b - 1$ zero-knowledge arguments of plaintext equality. One can most probably apply the proxy verifiable HPET protocol also to other protocols in an analogous manner.

Acknowledgements

This work was partially supported by the Finnish Defense Forces Research Institute of Technology. We would like to thank Yuval Ishai, Markus Jakobsson and Benny Pinkas for useful comments during various stages of writing this paper.

References

- [AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, Innsbruck, Austria, 6–10 May 2001. Springer-Verlag.
- [AJL03] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. Cryptographic Randomized Response Techniques. Technical Report 2003/027, International Association for Cryptologic Research, February 10 2003.
- [Bou00] Fabrice Boudot. Efficient Proofs that a Committed Number Lies in an Interval. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444, Bruges, Belgium, May 14–18 2000. Springer-Verlag.
- [BST01] Fabrice Boudot, Berry Schoenmakers, and Jacques Traoré. A Fair and Efficient Solution to the Socialist Millionaires’ Problem. *Discrete Applied Mathematics*, 111(1–2):23–36, 2001.
- [CC00] Christian Cachin and Jan Camenisch. Optimistic Fair Secure Computation. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 93–111, Santa Barbara, USA, 20–24 August 2000. International Association for Cryptologic Research, Springer-Verlag.
- [CD97] Ronald Cramer and Ivan Damgård. Linear zero-knowledge – a note on efficient zero-knowledge proofs and arguments. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 436–445, 1997.
- [CGHN01] Dario Catalano, Rosario Gennaro, Nick Howgrave-Graham, and Phong Q. Nguyen. Paillier’s Cryptosystem Revisited. In *8th ACM Conference on Computer and Communications Security*, pages 206–214, Philadelphia, Pennsylvania, USA, 6–8 November 2001. ACM Press.
- [CvdGT95] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed Oblivious Transfer and Private Multi-Party Computation. In Don Coppersmith, editor, *Advances in Cryptology — CRYPTO ’95*, volume 963 of *Lecture Notes in Computer Science*, pages 110–123, Santa Barbara, USA, 27–31 August 1995. Springer-Verlag.
- [DJ01] Ivan Damgård and Mads Jurik. A Generalisation, a Simplification and Some Applications of Paillier’s Probabilistic Public-Key System. In Kwangjo Kim, editor, *Public Key Cryptography ’2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, Korea, 13–15 February 2001. Springer-Verlag.
- [DJ03] Ivan Damgård and Mads Jurik. A Length-Flexible Threshold Cryptosystem with Applications. In Rei Safavi-Naini, editor, *The 8th Australasian Conference on Information Security and Privacy*, *Lecture Notes in Computer Science*, Wollongong, Australia, July 9–11 2003. Springer-Verlag. To appear.
- [El 84] Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, California, USA, 19–22 August 1984. Springer-Verlag, 1985.

- [FNW96] Ron Fagin, Moni Naor, and Peter Wrinkler. Comparing Information Without Leaking It. *Communications of the ACM*, 39:77–85, May 1996.
- [Kil88] Joe Kilian. Founding Cryptography on Oblivious Transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, Illinois, USA, 2–4 May 1988. ACM Press.
- [LAN02] Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure Vickrey Auctions without Threshold Trust. In Matt Blaze, editor, *Financial Cryptography — Sixth International Conference*, volume 2357 of *Lecture Notes in Computer Science*, pages 87–101, Southhampton Beach, Bermuda, March 11–14 2002. Springer-Verlag.
- [Lip03] Helger Lipmaa. On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In Chi Sung Lai, editor, *Advances on Cryptology — ASIACRYPT 2003*, *Lecture Notes in Computer Science*, Taipei, Taiwan, November 30–December 4 2003. Springer-Verlag. This volume.
- [NP99] Moni Naor and Benny Pinkas. Oblivious Transfer and Polynomial Evaluation. In *Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computing*, pages 245–254, Atlanta, Georgia, USA, 1–4 May 1999. ACM Press.
- [NP01] Moni Naor and Benny Pinkas. Efficient Oblivious Transfer Protocols. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457, Washington, DC, USA, January 7–9 2001. ACM Press.
- [NPS99] Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy Preserving Auctions and Mechanism Design. In *The 1st ACM Conference on Electronic Commerce*, Denver, Colorado, November 1999.
- [NS98] David Naccache and Jacques Stern. A New Public Key Cryptosystem Based on Higher Residues. In *5th ACM Conference on Computer and Communications Security*, pages 59–66, San Francisco, CA, USA, 3–5 November 1998. ACM Press.
- [OU98] Tatsuki Okamoto and Shigenori Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 308–318, Helsinki, Finland, May 31 – June 4 1998. Springer-Verlag.
- [Pai99] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, 2–6 May 1999. Springer-Verlag.
- [Ped91] Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, California, USA, August 11–15 1991. Springer-Verlag, 1992.
- [Tze02] Wen-Guey Tzeng. Efficient 1-Out-n Oblivious Transfer Schemes. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography '2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 159–171, Paris, France, February 12–14 2002. Springer-Verlag.

Generalized Powering Functions and Their Application to Digital Signatures

Hisayoshi Sato¹, Tsuyoshi Takagi², Satoru Tezuka¹, and Kazuo Takaragi¹

¹ Hitachi, Ltd., Systems Development Laboratory
292, Yoshida-cho, Totsuka-ku, Yokohama, 244-0817, Japan
{hisato,tezuka,takara}@sdl.hitachi.co.jp

² Technische Universität Darmstadt, Fachbereich Informatik
Alexanderstr.10, D-64283 Darmstadt, Germany
takagi@informatik.tu-darmstadt.de

Abstract. This paper investigates some modular powering functions suitable for cryptography. It is well known that the Rabin encryption function is a 4-to-1 mapping and breaking its one-wayness is secure under the factoring assumption. The previously reported encryption schemes using a powering function are variants of either the 4-to-1 mapping or higher n -to-1 mapping, where $n > 4$. In this paper, we propose an optimized powering function that is a 3-to-1 mapping using a p^2q -type modulus. The one-wayness of the proposed powering function is as hard as the infeasibility of the factoring problem. We present an efficient algorithm for computing the decryption for a p^2q -type modulus, which requires neither modular inversion nor division. Moreover, we construct new provably secure digital signatures as an application of the optimized functions. In order to achieve provable security in the random oracle model, we usually randomize a message using random hashing or padding. However, we have to compute the randomization again if the randomized message is a non-cubic residue element — it is inefficient for long messages. We propose an algorithm that can deterministically find the unique cubic residue element for a randomly chosen element.

Keywords: factoring, RSA, modular powering function, digital signature.

1 Introduction

Modular powering functions with composite moduli play an important role in cryptography. The RSA cryptosystem [15] and its variations [2,3] use one-to-one modular powering functions (permutations) as primitives. The Rabin cryptosystem [14] and its variants such as Williams' scheme [19] and Kurosawa et al.'s schemes [8,9] are composed of modular squaring functions (4-to-1 mapping). The other encryption schemes using powering functions [16],[10],[20],[21] utilize n -to-1 mappings ($n \geq 4$). Although the types of moduli for these functions are various, the following types are mainly used: pq , pqr and p^2q (*e.g.* [5],[7],[13],[18]), where

p, q, r are distinct prime numbers. The pqr -type modulus can efficiently compute its decryption using the Chinese remainder theorem [13], and the p^2q -type modulus can achieve faster decryption through the addition of Hensel lifting [7],[18].

These various kinds of functions have advantages and disadvantages. In cryptographic use, we expect that these functions will be proven to be one-way under some reliable assumptions such as the infeasibility of factoring large composite numbers. In view of this, strictly speaking, computational equivalence between one-wayness and the infeasibility of factoring is not proven for RSA functions. On the other hand, it is proven that Rabin functions are one-way under the factoring assumptions. However, for pq and p^2q -type moduli, these functions are 4-to-1, where four is the cardinality of the kernel (for pqr -type, the functions are 8-to-1), and this causes some inconvenience in cryptography such as non-uniqueness in decryption. In avoiding this, additional treatment is required for decryption or efficiency is decreased, and thus a smaller kernel would better suit for our purposes. Moreover, for a p^2q -type modulus, the conventional methods ([7],[18]) require modular inverses and integer divisions to be calculated. Even though these operations are fast in software, they are relatively expensive in hardware, especially for smartcards that are not equipped with coprocessors to calculate these inverses and divisions.

In this paper, we investigate optimized modular powering functions whose one-wayness can be proven secure under factoring assumptions. We deal with the general powering function $f(x) = x^e \bmod n$ for modulus $n = p^d q$. We show some criteria related to parameters g, d, p, q , which determine the number of pre-images of $f(x)$. We conclude that the optimal encryption of our proposed scheme is a 3-to-1 mapping with the p^2q -type modulus. Moreover, we propose an efficient algorithm to calculate the preimages of a p^2q -type modulus, which needs neither modular inversion nor division of integers. Moreover, as an application of these optimized functions, we construct new provably secure digital signatures using these functions in the random oracle model. In order to achieve the security in the random oracle model, we randomize a message using a random hashing or padding. If the randomized message is a non-cubic element, we have to randomize it again before the primitive computation — it is inefficient for long messages. In this paper, we propose an algorithm, with which the three possible kernel elements can easily be distinguished by a non-cubic residue element. This trick was initially proposed by Kurosawa et al. for the Rabin signature scheme [9]. Finally, we estimate the efficiency of the proposed signature scheme in contrast with other conventional signature schemes. The decryption of the proposed scheme with a p^2q -type modulus is about 1.7 time faster than that of the Multi-Prime RSA with a pqr -type modulus of the same size.

This paper is organized as follows: In Section 2, we discuss the proposed primitives and propose an optimal powering function. Section 3 discuss our construction of digital signature schemes based on the optimal powering function. Section 4 concludes the paper with a few closing remarks.

2 Proposed Primitives

In this section, we first study generalized powering functions with respect to security, efficiency and convenience for cryptography, and we then propose a new type of function that has not been applied to cryptography, that is, the conditions for prime factors of the modulus are asymmetric. In the following, because of efficiency, we concentrate on moduli of the powering functions that have two distinct prime factors (*cf.* Section 3.4).

2.1 Generalized Powering Functions

Let us recall the general properties of powering functions over some finite rings.

We deal with a modulus $N = p^d q$, where p and q are distinct odd prime numbers and $d \geq 1$ a positive integer¹. Let us denote the g -th power map on the multiplicative group \mathbb{Z}_N^* by

$$f = f_{N,g} : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*, \quad f(x) = x^g \bmod N. \quad (1)$$

Note that f is a $g_p g_q$ -to-one map, and the image of f is equal to $(\mathbb{Z}_N^*)^g = (\mathbb{Z}_{p^d}^*)^{g_p} \times (\mathbb{Z}_q^*)^{g_q}$, where $g_p = \gcd(g, p-1)$ and $g_q = \gcd(g, q-1)$ for an integer $1 < g < \min(p-1, q-1)$. Let y be an element in the image of f , then the preimage of y by f is the set given by $f^{-1}(y) = \{x' \in \mathbb{Z}_N^* \mid x'^g = y\}$, which consists of $g_p g_q$ elements. We choose $g = 2$ for the Rabin cryptosystem, so that there are four ambiguities for the preimage of map f due to $g_p = g_q = 2$.

We denote the isomorphism by the Chinese remainder theorem by ϕ :

$$\phi = \phi_{p^d, q} : \mathbb{Z}_{p^d} \times \mathbb{Z}_q \xrightarrow{\sim} \mathbb{Z}_N.$$

As is well known, the multiplicative group \mathbb{Z}_p^* is a cyclic group of order $p-1$. For an integer $t \geq 1$, let $Z_{p,t}$ be the subgroup of elements in \mathbb{Z}_p^* whose order divides t : $Z_{p,t} = \{a \in \mathbb{Z}_p^* \mid a^t = 1\}$.

We consider the g -th power map $f_{p,g}$ on \mathbb{Z}_p^* . It can easily be seen that the following sequence is exact²: $1 \rightarrow Z_{p,g} \hookrightarrow \mathbb{Z}_p^* \xrightarrow{f_{p,g}} (\mathbb{Z}_p^*)^g \rightarrow 1$. Moreover, we have $(\mathbb{Z}_p^*)^g = ((\mathbb{Z}_p^*)^{g_p})^{g/g_p}$, and the order of $(\mathbb{Z}_p^*)^{g_p}$ is $(p-1)/g_p$, in particular, it is prime to g/g_p , thus it holds that $((\mathbb{Z}_p^*)^{g_p})^{g/g_p} = (\mathbb{Z}_p^*)^{g_p}$. Hence, we have $\#Z_{p,g} = (p-1)/\#(\mathbb{Z}_p^*)^{g_p} = g_p$, and from the uniqueness of the subgroup of order g_p in \mathbb{Z}_p^* , letting $\zeta_{p,g}$ be a primitive g_p -th root of unity, we have $Z_{p,g} = Z_{p,g_p} = \langle \zeta_{p,g} \rangle$, that is, the subgroup of g -th roots of unity is equal to that of g_p -th roots of unity. Let $\chi_{p,g} = f_{p, \frac{p-1}{g_p}}$ be the $(p-1)/g_p$ -th power map on \mathbb{Z}_p^* :

$$\chi_{p,g} : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*, \quad \chi_{p,g}(x) = x^{\frac{p-1}{g_p}} \bmod p,$$

¹ Boneh et al. proposed a polynomial time algorithm for factoring $p^d q$ for large d [4].

The exponent d in this paper is very small, so that their algorithm is not effective.

² $A \xrightarrow{f} B \xrightarrow{g} C$ is said to be exact if the image of f is equal to the kernel of g .

then, due to the above arguments, the following sequence is exact: $1 \rightarrow (\mathbb{Z}_p^*)^g = (\mathbb{Z}_p^*)^{g_p} \hookrightarrow \mathbb{Z}_p^* \xrightarrow{\chi_{p,g}} \langle \zeta_{p,g} \rangle \rightarrow 1$. In other words, we have the following.

Lemma 1. *For any $x \in \mathbb{Z}_p^*$, we have $\chi_{p,g}(x) \in \langle \zeta_{p,g} \rangle$, and x is a g -th power residue (i.e. $x \in (\mathbb{Z}_p^*)^g$) if and only if $\chi_{p,g}(x) = 1$.*

For $(\mathbb{Z}_{p^d})^*$, there exists a decomposition $(\mathbb{Z}_{p^d})^* \cong (\mathbb{Z}_p)^* \times \mathbb{Z}_{p^{d-1}}$. Since $\gcd(g, p) = 1$, regarding \mathbb{Z}_p^* as a subgroup of $\mathbb{Z}_{p^d}^*$, we have $\mathbb{Z}_{p^d,g} = \mathbb{Z}_{p,g}$, and $a \in \mathbb{Z}_{p^d}^*$ is a g -th power residue if and only if $a \bmod p$ satisfies the condition in Lemma 1. For the prime q , the situation is similar, let $\mathbb{Z}_{q,g} = \langle \zeta_{q,g} \rangle$, then by the Chinese remainder theorem, the set $Z_{N,g}$ of all g -th roots of unity in \mathbb{Z}_N^* can be written as $Z_{N,g} = \phi(\mathbb{Z}_{p,g}, \mathbb{Z}_{q,g}) = \{ \phi(\zeta_{p,g}^i, \zeta_{q,g}^j) \mid 0 \leq i < g_p, 0 \leq j < g_q \}$. Since $\zeta_{p,g}$ and $\zeta_{q,g}$ are easily calculated (e.g. $\zeta_{p,g} = x^{(p-1)/g_p}$ for some x), and by Lemma 2, $Z_{N,g}$ is easily obtained. Especially, we have $\#Z_{N,g} = g_p \cdot g_q$. Putting $g_{p,q} = \text{lcm}(g_p, g_q)$, it can easily be seen that $Z_{N,g} = Z_{N,g_{p,q}}$.

Consequently, for $y \in (\mathbb{Z}_N^*)^g$, let $x \in f^{-1}(y)$ be a preimage of $y : x^g = y$, then the preimage $f^{-1}(y)$ of y by f can easily be calculated by the following:

$$f^{-1}(y) = \{ x \cdot \phi(\zeta_{p,g}^i, \zeta_{q,g}^j) \mid 0 \leq i < g_p, 0 \leq j < g_q \}, \quad (2)$$

2.2 Security of Generalized Powering Functions

We will now consider computational equivalence between the factoring problem on N and the one-wayness of function f , when f is not injective.

For each divisor e of g , define the set of primes which satisfy the conditions in Lemma 1 as follows: $\mathcal{P}_e = \{p : \text{prime} \mid \gcd(g, p-1) = e, \gcd(g, (p-1)/e) = 1\}$.

Let $\text{Div}(g)$ be the set of all divisors of g . For a non-empty set $\mathcal{D} \subset \text{Div}(g)$, we put $\mathcal{P}_{\mathcal{D}} = \bigcup_{e \in \mathcal{D}} \mathcal{P}_e$. We fix integers $d \geq 1$, $g \geq 2$, and non-empty sets $\mathcal{D}_1, \mathcal{D}_2 \subset \text{Div}(g)$, $\mathcal{D}_1 \cup \mathcal{D}_2 \neq \{1\}$ (namely, one of these contains divisors of g besides 1). For these, let a instance generator \mathcal{G}_0 be a probabilistic polynomial time algorithm such that $\mathcal{G}_0(1^k) \rightarrow N$, where $N = p^d q$, $|N| = k$, $p \in \mathcal{P}_{\mathcal{D}_1}$, $q \in \mathcal{P}_{\mathcal{D}_2}$, $|p| \approx |q|$ (In the case $1 \in \mathcal{D}_1 \cap \mathcal{D}_2$, we also assume that $(p, q) \notin \mathcal{P}_1 \times \mathcal{P}_1$). Using these notations, we define the factoring problem and its infeasibility.

Definition 1. *The integer factoring problem is a problem which for given d , g , \mathcal{D}_1 , \mathcal{D}_2 , and $N \xleftarrow{R} \mathcal{G}_0(1^k)$, finds the factors (p, q) of N . The integer factoring problem is said to be infeasible if for any probabilistic polynomial time (PPT) algorithm A , any constant c and all sufficiently large k ,*

$$\Pr \left[A(1^k, N, d, g, \mathcal{D}_1, \mathcal{D}_2) = (p, q) \mid N \xleftarrow{R} \mathcal{G}_0(1^k) \right] < \frac{1}{k^c}.$$

Definition 2. *A integer factoring PPT algorithm A is said to (t, ϵ) -break $N \xleftarrow{R} \mathcal{G}_0(1^k)$ if for any $k \in \mathbb{N}$, after at most $t(k)$ processing time, it factors N with probability at least $\epsilon(k)$. The set of outputs of \mathcal{G}_0 is (t, ϵ) -secure if there exists no integer factoring PPT algorithm which (t, ϵ) -breaks.*

We next define the one-wayness for the functions defined in Section 2.1 as follows:

Definition 3. *The notations $d, g, \mathcal{D}_1, \mathcal{D}_2$ and \mathcal{G}_0 are the same as in Definition 1. The powering function f is said to be one-way if for any PPT algorithm A , for any constant c and any sufficiently large k ,*

$$\text{Adv}(A) = \Pr \left[A(1^k, N, d, g, \mathcal{D}_1, \mathcal{D}_2, y) = x' \in f^{-1}(y) \mid \begin{array}{l} N \xleftarrow{R} \mathcal{G}_0(1^k); \\ x \xleftarrow{R} \mathbb{Z}_N^*; \\ y \xleftarrow{R} f(x) \end{array} \right] < \frac{1}{k^c}.$$

Definition 4. *A PPT algorithm A is said to (t, ϵ) -break f if for any $k \in \mathbb{N}$, after at most $t(k)$ processing time, it calculates a preimage of f with probability at least $\epsilon(k)$. f is (t, ϵ) -secure if there exists no PPT algorithm which (t, ϵ) -breaks.*

Let φ be the Euler totient function. Under these definitions, we can prove the following theorem, whose proof can be found in Appendix A.

Theorem 1. *Fix integers $d \geq 1$ and $g \geq 2$, and assume that all divisors of g can be efficiently computed. Fix non-empty sets $\mathcal{D}_1, \mathcal{D}_2 \subset \text{Div}(g)$, $\mathcal{D}_1 \cup \mathcal{D}_2 \neq \{1\}$, and for any $e_1 \in \mathcal{D}_1$ and $e_2 \in \mathcal{D}_2$, assume that $\{\sum_{e \mid \gcd(e_1, e_2)} \varphi(e)^2\} / e_1 e_2$ is small. Moreover, we put*

$$\tau = \min_{e_1 \in \mathcal{D}_1, e_2 \in \mathcal{D}_2} \left\{ 1 - \frac{\sum_{e \mid \gcd(e_1, e_2)} \varphi(e)^2}{e_1 e_2} \right\}.$$

(Notice that by the assumptions, τ is close to 1) Let \mathcal{G}_0 be the instance generator for the above parameters, and $N \xleftarrow{R} \mathcal{G}_0(1^k)$. If the integer factoring problem for N is infeasible, then the function $f : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$, $f(x) = x^g$ is one-way. More precisely, if the outputs of \mathcal{G}_0 are (t_I, ϵ_I) -secure, then f is (t_f, ϵ_f) -secure, where

$$t_I(k) = t_f(k) + O(k^3), \quad \epsilon_I(k) = \tau \epsilon_f(k).$$

According to the argument in Section 2.3, the inverse of f can be calculated using the factors of N , hence, together with the argument in this section, it is proven that the equivalence between the infeasibility of the factoring problem on N and the one-wayness of f .

2.3 Efficient Decryption Algorithms

In this section, we consider an efficient algorithm that can be used to calculate the preimage of the powering function considered in Section 2.1, (1) when the prime factors p and q are known. In the case $d > 1$, the conventional method

(e.g. [7],[18]) needs the modular inverse to be calculated. We now propose an algorithm that does not need the modular inverse to be calculated under some conditions on p , q and g . The proofs for the following are in Appendix B. The notations p , q , d , N and g are the same as in Section 2.1. Let $z = p^{-1} \bmod q$. For $y \in (\mathbb{Z}_N^*)^g$, we put $y_p = y \bmod p$, $y_q = y \bmod q$ and $y_i = y \bmod p^i q$ ($1 \leq i \leq d$). Moreover, let x_p be a g -th root of y_p in \mathbb{Z}_p^* , that is $x_p^g = y_p \pmod{p}$. Similarly, let x_q be a g -th root of y_q : $x_q^g = y_q \bmod q$. Then, by the Chinese remainder theorem, a g -th root of y modulo pq is given by the following lemma.

Lemma 2. *By the isomorphism $\mathbb{Z}_p \times \mathbb{Z}_q \xrightarrow{\sim} \mathbb{Z}_{pq}$, $(x_p, x_q) \in \mathbb{Z}_p \times \mathbb{Z}_q$ corresponds to an element $x_1 \in \mathbb{Z}_{pq}$, which is given by $x_1 = x_p + p((x_q - x_p)z \bmod q)$, and x_1 is a g -th root of $y_1 (= y \bmod pq)$.*

By using the g -th roots of $y \in (\mathbb{Z}_N^*)^g$ in modulus p , q and pq , we can calculate the g -th root of y in the high-power modulus ($p^i q$, $i = 2, 3, \dots, d$) as we will see in the following.

Lemma 3. *The notations are the same as in the above. Let $\eta_y = (gx_p^{g-1})^{-1} \bmod p$ and x_i ($1 \leq i < d$) be a g -th root of y_i (modulo $p^i q$) such that $x_i \equiv x_p \pmod{p}$. Then a g -th root x_{i+1} of y_{i+1} modulo $p^{i+1} q$ such that $x_{i+1} \equiv x_p \bmod p$ is given by $x_{i+1} = x_i + \eta_y(y_{i+1} - x_i^g) \bmod p^{i+1} q$.*

Though it needs to calculate a modular inverse modulo p for η_y , under some condition, we can have η_y efficiently.

Lemma 4. *Assume that there exists some integer $0 \leq \alpha < p-1$ depending only on p and g such that $x_p = y_p^\alpha \bmod p$, then we have $(x_p^{g-1})^{-1} \bmod p = y_p^{\alpha-1} \bmod p$.*

This follows from $y_p^{\alpha-1} \cdot x_p^{g-1} = (x_p \cdot y_p^{-1}) \cdot (y_p \cdot x_p^{-1}) = 1 \pmod{p}$. Thus, once we obtain $(x_p^{g-1})^{-1} \bmod p$, precalculating $g^{-1} \bmod p$, we have $\eta_y = g^{-1} (x_p^{g-1})^{-1} \bmod p$ by single modular multiplication.

Let us next consider the conditions in Lemma 4. That is, let us consider the relation between p and g so that there exists an integer α which depends only on p and g , such that for any $a \in (\mathbb{Z}_p^*)^g$, a^α is a g -th root of a ($(a^\alpha)^g = a$).

Proposition 1. *Let p be a prime, $1 < g < p-1$ be an integer. Then there exists an integer $\alpha = \alpha(p, g)$ ($1 \leq \alpha < p-1$) which depends only on p and g , and satisfies $(a^\alpha)^g = a$ for any $a \in (\mathbb{Z}_p^*)^g$ if and only if $\gcd(g, (p-1)/g_0) = 1$ where $g_0 = \gcd(g, p-1)$. Here, α is given by*

$$\alpha = \alpha(p, g) = (1 + u\{(p-1)/g_0\})/g, \quad \text{where } u = (-(p-1)/g_0)^{-1} \bmod g.$$

Using the above discussion, if for p and q , g satisfies the conditions in Lemma 1, we have an efficient algorithm which calculates a g -th root (Note that using a g -th root, all g -th root are given by (2)).

Corollary 1. *Let p, q be prime integers. Let $d > 1$ be an integer, and $N = p^d q$. Let $g > 1$ be an integer which satisfies $g < \min(p - 1, q - 1)$ and $\gcd(g, (p - 1)/\gcd(g, p - 1)) = \gcd(g, (q - 1)/\gcd(g, q - 1)) = 1$. Moreover, let $z = p^{-1} \bmod q$, $\gamma = g^{-1} \bmod p$. For any $y \in (\mathbb{Z}_N^*)^g$, put $y_p = y \bmod p$, $y_q = y \bmod q$, $y_i = y \bmod p^i q$ ($1 \leq i < d$), $y_d = y$. Then by calculating*

$$\begin{aligned} x_0 &= y_p^{\alpha(p,g)-1} \bmod p, & x_p &= y_p x_0 \bmod p, \\ \eta &= w x_0 \bmod p, & x_q &= y_q^{\alpha(q,g)} \bmod q, \\ x_1 &= x_p + p((x_q - x_p)z \bmod q), \end{aligned}$$

and for $i = 2, 3, \dots, d$, $x_i = x_{i-1} + \eta(y_i - x_{i-1}^g) \bmod p^i q$, we have that $x := x_d$ is a g -th root of y ($x^g = y \bmod N$).

2.4 Choices of Cryptographically Suitable Powering Functions

We will discuss the optimal choice of parameters g, g_p, g_q and d suitable for cryptography in the following.

Efficiency must be considered, when we apply powering functions that can be proven to be one-way under the assumption of infeasibility of the integer factoring problem to cryptosystems. The cost of calculating the image will be lower if the powering index is smaller. Moreover, if the number of preimages is larger, then there will be some inconvenience as previously mentioned.

Table 1. Parameters for $2 \leq g \leq 8$.

g	g_p	g_q	$g_p \cdot g_q$	τ
2	2	2	4	.500
3	1	3	3	.667
	3	3	9	.444
4	2	4	8	.750
	4	4	16	.625
5	1	5	5	.800
	5	5	25	.320

g	g_p	g_q	$g_p \cdot g_q$	τ
6	2	6	12	.833
	6	6	36	.722
7	1	7	7	.857
	7	7	49	.245
8	2	8	16	.875
	4	8	32	.813
	8	8	64	.656

Table 1 shows all possibilities of g_p, g_q for relatively small g 's. Note that cases where p and q are replaced have been omitted, and for even g , the parities of g_p and g_q (even or odd) coincide. Note also that the case $(g, g_p, g_q) = (2, 2, 2)$ (and $d = 1$) corresponds to the Rabin function. The value $g_p \cdot g_q$ indicates that the g -th power function f is a $(g_p \cdot g_q)$ -to-1 mapping and is desired to be small. τ is the constant coefficient appearing in the reduction probability of Theorem 1 (See also Appendix A for more details). Although a larger value is desired, it is sufficient if it is greater than 0.5. From this table, we can conclude that the case $(g, g_p, g_q) = (3, 1, 3)$ (or $(g, g_p, g_q) = (3, 3, 1)$), that has smallest $g_p \cdot g_q$, is optimized for cryptosystems (ratio is 0.667 and sufficiently large). Moreover, as we will see in Section 3.4, the $p^d q$ -type modulus ($d \geq 2$) makes the preimage calculation more efficient using the proposed algorithm in Section 2.3.

Table 2. (3,1,3)-type and other typical functions.

type (g, g_p, g_q)	map $g_p g_q : 1$	condition for p, q	$p^d q$ -type modulus		assumption for one-wayness
			$d = 1$	$d \geq 2$	
$(e, 1, 1)$	1:1	symmetric	RSA	—	RSA
$(3, 1, 3)$	3:1	asymmetric	—	F	IF
$(2, 2, 2)$	4:1	symmetric	Rabin	HIME	IF

Thus letting $F (= f_{N,3})$ be the (3,1,3)-type function with the $p^d q$ -type modulus ($d \geq 2$), we can conclude that F is most suitable for cryptography. Table 2 sums up the positions of F and other typical functions including RSA functions. In the table, IF stands for integer factorization.

3 Application to Digital Signatures

As cryptographic applications of the arguments in the previous sections, we propose digital signature schemes using the cubic function considered in Section 2.4 and ensure the advantages of the proposed cubic function, especially in terms of efficiency (Section 3.4).

3.1 Basic Notation

We start by recalling the basic notion of digital signature schemes according to [3,6,12].

Definition 5. A signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ is defined as follows:

The key generation algorithm \mathcal{G} is a PPT algorithm which has input 1^k and outputs a pair of matching public and secret keys (pk, sk) .

The signature generation algorithm \mathcal{S} takes a message M to be signed and public and secret keys (pk, sk) , and outputs a signature $x = \mathcal{S}_{pk,sk}(M)$.

The signature verification algorithm \mathcal{V} takes a message M , a candidate signature x' and public key pk , and outputs a bit $\mathcal{V}_{pk}(M, x')$, equal to 1 if the signature is accepted and 0 otherwise. We require that if $x = \mathcal{S}_{pk,sk}$, then $\mathcal{V}_{pk}(M, x') = 1$.

On the security for signatures, we only deal with existential unforgeability under an adoptive chosen message attack which is the strongest notion([3,6]). In this scenario, a forger of a signature can dynamically obtain signatures of messages of his choice and attempts to output a valid signature, where a pair of message and signature (M, x) is said to be a valid forgery if $\mathcal{V}_{pk}(M, x) = 1$ and the signature of M was never requested by the forger.

Most signature schemes use hash functions, and the security is proven under random oracle models, that is the models which is appropriately replaced the hash functions with random oracles([1]). In these models, forgers are allowed to access to random oracles. The resistance against these attacks is defined as follows:

Definition 6. A forger \mathcal{A} (a PPT algorithm) is said to (t, q_h, q_s, ϵ) -breaks the signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ if after at most $q_h(k)$ queries to the hash oracles, $q_s(k)$ signature queries and $t(k)$ processing time, it outputs a valid forgery with probability at least $\epsilon(k)$ (for any $k \in \mathbb{N}$). A signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ is (t, q_h, q_s, ϵ) -secure if there exists no forger who (t, q_h, q_s, ϵ) -breaks the scheme.

3.2 Proposed Signature Scheme: Scheme 1

We now propose new signatures constructed with the (3,1,3)-type cubic residue function F in Section 2.4 (in case $d = 2$). These are proven to be secure under the assumption of integer factoring infeasibility.

First, we will consider the (full domain) hash & sign (F -FDHS) signature which is most fundamental. Fix an integer $a > 1$ (regard a as a system parameter).

Key Generation

Generate randomly same length distinct prime numbers p and q such that $p \equiv 2 \pmod{3}$, $q \equiv 4$ or $7 \pmod{9}$, and choose a non-cubic residue a modulo q , put $N = p^2q$. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$ be a hash function. Then output the public key (N, H) and the secret key (p, q) (a is open to public as a system parameter).

Signature Generation

1. For a message M , calculate $w = H(M)$.
2. Let y be one of w, aw, a^2w which is a cubic residue.
3. Calculate a cubic root x of y ($x \in F^{-1}(y)$).
4. Output x and end.

Signature Verification

1. For the message M , calculate $w' = H(M)$.
2. Calculate $y' = x^3 \pmod{N}$.
3. If y' coincides one of w', aw', a^2w' , then output 1, else output 0 and end.

Remark 1. Note that from Lemma 1, we can easily seen that one of w, aw or a^2w is a cubic residue, and we can determine this by calculating $\chi_{p,3}$ (this function is also a powering function). Therefore, we do not have to recompute the hash value $H(M)$, and for a given message m we can uniquely generate the signature x of m . Kurosawa et al. proposed a similar technique for the Rabin signature [9].

We can prove that F -FDHS is secure over the random oracle model (the hash function H is replaced to the random oracles) under the assumption of integer factoring infeasibility. The proof is basically similar to that of [9].

For the fixed a and a positive integer k , let

$$\mathcal{N}_k = \left\{ N = p^2q \left| \begin{array}{l} p, q : \text{primes, } |p| = |q| = k, \ p \pmod{3} = 2, \\ q \pmod{9} = 4 \text{ or } 7, \ a \in \mathbb{Z}_q^* \setminus (\mathbb{Z}_q^*)^3 \end{array} \right. \right\},$$

and put $\mathcal{N} = \bigcup_k \mathcal{N}_k$. Then we can prove the following theorem, whose proof can be found in Appendix C.

Theorem 2. *If \mathcal{N} is (t_I, ϵ_I) -secure, then F -FDHS is (t, q_H, q_s, ϵ) -secure, where,*

$$t_I(k) = t(k) + (q_H + q_s + 2)O(k^3), \quad \epsilon_I(k) = (2/3)\epsilon(k).$$

In the following, combining the idea in Lemma 1 and Corollary 1, we propose an efficient algorithm, denoted by Φ , which for given $w \in \mathbb{Z}_N^*$, determines which of w, aw or a^2w is a cubic residue, and then calculates its cubic root. The validity of the algorithm can be found in Appendix D.

Let $\gamma = (p+1)/3$ and $z = p^{-1} \bmod q$. Let $\beta_p = (2p-4)/3$ and $\beta_q = (2q-8)/9$, $\zeta = a^{(q-1)/3} \bmod q$ if $q \equiv 4 \bmod 9$, $\beta_q = (q-7)/9$, $\zeta = a^{(2(q-1))/3} \bmod q$ if $q \equiv 7 \bmod 9$. Finally, let $b = a^{\beta_q+1} \bmod q$.

Algorithm Φ

Input: $N, a, p, q, \beta_p, \beta_q, b, \zeta, z, \gamma$ and $w \in \mathbb{Z}_N^*$.

Output: $x \in \mathbb{Z}_N^*$ s.t. $x^3 \bmod N \in \{w, aw \bmod N, a^2w \bmod N\}$.

Step 1. Check the cubic residuosity modulo q and calculate a cubic root

Step 1.1. $w_q = w \bmod q$.

Step 1.2. $w_1 = w_q^{\beta_q} \bmod q$.

Step 1.3. $x_q = w_1 w_q \bmod q$.

Step 1.4. $w_3 = w_1 x_q^2 \bmod q$.

Step 1.5. if $w_3 \neq 1$ then

Step 1.5.1. Set $x_q \leftarrow b_q x_q \bmod q$, $w \leftarrow aw \bmod N$.

Step 1.5.2. If $w_3 \neq \zeta$ then set $x_q \leftarrow b_q x_q \bmod q$, $w \leftarrow aw \bmod N$.

Step 2. Calculate a cubic root modulo p

Step 2.1. $w_p = w \bmod p$.

Step 2.2. $x_0 = w_p^{\beta_p} \bmod p$.

Step 2.3. $x_p = w_p x_0 \bmod p$.

Step 2.4. $\eta = \gamma x_0 \bmod p$.

Step 3. $x_1 = x_p + p((x_q - x_p)z \bmod q)$.

Step 4. $x = x_1 + \eta(w - x_1^3) \bmod N$.

3.3 Other Constructions: Schemes 2 and 3

In the following, we present two additional constructions of digital signatures based on the generalized powering function.

Scheme 2. Let us consider a scheme F -2HS (2-hash and sign) that has been slightly changed from scheme 1 : F -FDHS. Similarly, fix an integer $a > 1$ and let k_1, k_2 be positive integers such that $k_1 + k_2 < |N|$ (modulus length), and regard these as system parameters in addition to a in scheme 1. The key generation is the same as for scheme 1 except for letting $H : \{0, 1\}^* \rightarrow \{0, 1\}^{k_1}$ and $G : \{0, 1\}^{k_1} \rightarrow \{0, 1\}^{k_2}$ be hash functions (H : compressor, G : generator). The public key is (N, H, G) , and the secret key is (p, q) .

Signature Generation

1. For a message M , calculate $w_1 = H(M)$, $w_2 = G(w_1)$ and let $w = w_1 || w_2$.
2. Let y be one of w , aw , a^2w which is a cubic residue.
3. Calculate a cubic root x of y ($x \in F^{-1}(y)$).
4. Output x and end.

Signature Verification

1. For M , calculate $w'_1 = H(M)$, $w'_2 = G(w'_1)$ and let $w' = w'_1 || w'_2$.
2. Calculate $y' = x^3 \bmod N$.
3. If y' coincides one of w' , aw' , a^2w' , then output 1, else output 0 and end.

This scheme can also be proven to be secure over the random oracle model (the hash functions H and G are replaced with random oracles) under the factoring assumption. Let \mathcal{N} be the same as in scheme 1. We then have the following:

Theorem 3. *If \mathcal{N} is (t_I, ϵ_I) -secure, then F -2HS is $(t, q_H, q_G, q_s, \epsilon)$ -secure, where*

$$t_I(k) = t(k) + (q_H + q_s + 2)O(k^3), \quad \epsilon_I(k) = (2/3)\epsilon(k).$$

This scheme is nothing more than a version of PSS([3]) without the random numbers part, and is not essentially different from scheme 1. However, with respect to implementation, we bother with the construction of hash functions with long output using some short output functions (e.g. [17]), and in most cases, it is inefficient when the hash function deals with very long messages.

Scheme 3. Finally, we will consider a message recovery signature scheme F -MR (message recovery) based on scheme 2 : F -2HS. For this scheme, the message length is restricted to $|M| = k_2$. The key generation and signature generation are the same as in scheme 2, except that we set $w_2 = G(w_1) \oplus M$ in Step 1 of the signature generation (Fig. 1). The signature verification and message recovery are as follows:

Signature Verification

1. Calculate $y' = x^3 \bmod N$.
2. For $i = 0, 1, 2$,
 - 2.1. Calculate $y_i = w_{1,i} || w_{2,i} = a_2^i y' \bmod N$ ($|w_{1,i}| = k_1$, $|w_{2,i}| = k_2$).
 - 2.2. Calculate $M_i = w_{2,i} \oplus G(w_{1,i})$.
3. If for some i , $w_{1,i} = H(M_i)$, then output 1 and M_i , else output 0 and end.

Similarly for this scheme, we can prove following:

Theorem 4. *If \mathcal{N} is (t_I, ϵ_I) -secure, then F -MR is $(t, q_H, q_G, q_s, \epsilon)$ -secure, where*

$$t_I(k) = t(k) + (q_H + q_s + 2)O(k^3), \quad \epsilon_I(k) = (2/3)\epsilon(k).$$

Similar to scheme 2, this scheme is nothing more than a version of PSS-R([3]) without the random number part, and this makes the message embedded in the signature longer than that of PSS-R.

As we have seen, the proposed schemes need no trial and error in hashing messages and in finding the cubic residue. This has a good effect on efficiency especially with huge messages. In what follows, we discuss the advantages in efficiency of the proposed schemes in detail.

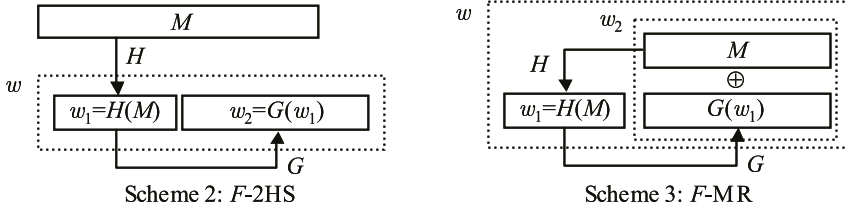


Fig. 1. Paddings for Schemes 2 and 3.

3.4 Efficiency Consideration

In this section, we estimate the efficiency of signature schemes 1,2 and 3 that are introduced in the previous sections.

The proposed schemes deal with the modulus of $N = p^2q$ ($|p| \approx |q|$). In order to fairly compare the proposed schemes with the RSA signature, we estimate the efficiency of a fast variant of RSA signature, namely Multi-Prime RSA with $N = pqr$ ($|p| \approx |q| \approx |r|$) [13]. We consider the efficiency of signature generation which has higher costs in comparison with signature verification. Note that Multi-prime (pqr -type) Rabin's scheme has the same efficiency as RSA signature in signature verification.

The efficiency of public-key cryptosystems and digital signatures is frequently estimated by the number of modular multiplications. Let us introduce the following notations to represent the amount of calculation. Let $\text{Mul}(t)$ denote the amount of calculation for an integer multiplication of t -bit integers. Similarly, let $\text{RMul}(t)$ be that for a modular multiplication with a t -bit modulus, and $\text{Red}(s, t)$ that for a reduction s -bit integer with a t -bit modulus. Also, let $\text{RP}(t)$ be the number of t -bit modulus modular multiplications for powering with a t -bit exponent.

In Schemes 1,2 and 3, the steps for checking the cubic residuosity and calculating a cubic root (function Φ in Section 3.2), comprise a large percentage of signature generation. Thus we consider the efficiency of Φ . Let ℓ be the bit-length of modulus $N = p^2q$ ($|p| \approx |q|$). Signature generation needs the following amount of calculation:

$$(8 + 2 \cdot \text{RP}(\ell/3)) \cdot \text{RMul}(\ell/3) + 3 \cdot \text{RMul}(\ell) + 2 \cdot \text{Red}(\ell, \ell/3) + \text{Mul}(\ell/3).$$

On the other hand, let ℓ be the bit-length of Multi-Prime RSA modulus $N = pqr$ ($|p| \approx |q| \approx |r|$), then for the generation of Multi-Prime RSA signature [13], it needs

$$(1 + 3 \cdot \text{RP}(\ell/3)) \cdot \text{RMul}(\ell/3) + 3 \cdot \text{Red}(\ell, \ell/3) + \text{Mul}(\ell/3).$$

We have approximately $\text{RMul}(t) \approx n^2 \cdot \text{RMul}(t/n)$, $\text{Mul}(t) \approx (1/2) \cdot \text{RMul}(t)$. Moreover, if we use the Montgomery method [11] for modular reduction, then we have $\text{Red}(t, t/n) \approx ((n+1)/n^2) \cdot \text{RMul}(t)$. We set a standard to the number of modular multiplication on 1024-bit modulus: $1 = \text{RMul}(1024)$. We also assume

that a t -bit modular multiplication costs $\xi(t) := (t/1024)^2$. Then the amount of calculation for the signature generation of the proposed and RSA schemes is

$$\begin{aligned}\text{Proposed schemes} &: \{29/6 + (2/9)\text{RP}(\ell/3)\} \xi(\ell), \\ \text{Multi-Prime RSA} &: \{3/2 + (1/3)\text{RP}(\ell/3)\} \xi(\ell).\end{aligned}$$

For modular powering, we adopt the basic binary method. If we assume that half the bits in the exponent are non-zero, then this method needs $3t/2$ modular multiplications with a t -bit modulus (where we also assume that modular squaring and modular multiplication have the same amount of calculation). Taking all this into account, the number of modular multiplications in the proposed schemes and the RSA signature and their ratio are as follows:

$$\begin{aligned}\text{Proposed Schemes} &: (29/6 + \ell/81) \xi(\ell), \quad \text{Multi-Prime RSA} : (3/2 + \ell/36) \xi(\ell), \\ \text{Multi-Prime RSA/Proposed scheme} &\approx 1.71.\end{aligned}$$

Thus, we can say that the proposed schemes are considerably more efficient in signature generation than Multi-Prime RSA signature. Similarly, we can see that the proposed schemes are three or more times more efficient than the pq -type RSA-CRT signature.

4 Summary

We studied modular powering functions suitable for cryptography. In particular, we proposed a 3-to-1 functions, which can be proven to be one-way under the factoring assumption. The three ambiguities of the kernel can easily be distinguished by a non-cubic residue element. For the $p^d q$ -type modulus ($d \geq 2$), we proposed a more efficient method of calculating preimages for these functions, which requires no modular inversion algorithm for Hensel lifting. Thus we can say that the proposed functions are optimized in terms of security and efficiency.

As cryptographic applications, we also proposed new digital signature schemes which utilize the new functions with $d = 2$. Finally, we showed that the proposed schemes are about 1.71 times more efficient than Multi-Prime RSA with the same length modulus (more than three times faster than the pq -type RSA-CRT signature).

References

1. M.BELLARE and P.ROGAWAY, Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, *Proceedings of the 1st ACM Conference on Computer and Communications Security*, ACM Press (1993), 62-73. Available at <http://www.cs.ucdavis.edu/~rogaway/papers/index.html>
2. M.BELLARE and P.ROGAWAY, Optimal asymmetric encryption – How to encrypt with RSA, *Advances in Cryptology – Eurocrypt'94*, LNCS 950, Springer-Verlag (1994), 92-111.

3. M.BELLARE and P.ROGAWAY, The exact security of digital signatures – How to sign with RSA and Rabin, *Advances in Cryptology – Eurocrypt’96*, LNCS 1070, Springer-Verlag (1996), 399-416.
4. D.BONEH, G.DURFEE and N.HOWGRAVE-GRAHAM, Factoring $N = p^r q$ for large r , *Advances in Cryptology – Crypto’99*, LNCS 1666, Springer-Verlag (1999), 326-337.
5. D.BONEH and H.SHACHAM, Fast Variants of RSA, *CRYPTOBYTES*, Vol.5, No.1, 2002, 1-9.
6. J.S.CORON, Optimal security proofs for PSS and other signature schemes, *Advances in Cryptology – EUROCRYPT 2002 Proceedings*, Springer-Verlag (2002), 272-287.
7. HIME(R) : High Performance Modular Squaring Based Public-Key Encryption (Revised Edition), Hitachi, Ltd., 2001.
8. K.KUROKAWA, T.ITO and M.TAKEUCHI, Public Key cryptosystem using a reciprocal number with the same intractability as factoring a large number, *Cryptologia*, 12 (1988), 225-233.
9. K.KUROKAWA and W.OGATA, Efficient Rabin-type Digital Signature Scheme, *Designs, Codes and Cryptography* 16 (1999), 53-64.
10. J.H.LOXTON, AND D.S.P.KHOO, G.J.BIRD AND J.SEBERRY, A cubic RSA code equivalent to factorization, *Journal of Cryptology*, 5, (1992), 139-150.
11. P.L.MONTGOMERY, Modular multiplication without trial division, *Math. Comp.*, 44 (1985), 519-521.
12. D.POINTCHEVAL and J.STERN, Security proofs for signature schemes, *Advances in Cryptology – Eurocrypt’96*, LNCS 1070, Springer-Verlag (1996), 399-416.
13. Public-Key Cryptography Standards, PKCS # 1, Amendment 1: Multi-Prime RSA, RSA Laboratories.
14. M.O.RABIN, Digitalized signatures and public key cryptosystems as intractable as factorization, Technical Report, MIT/LCS/TR-212, MIT, Cambridge, MA (1979).
15. R.L.RIVEST, A.SHAMIR AND L.ADLEMAN, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, Vol.21, No.2 (1978), 120-126.
16. R.SCHIEDLER, A Public-Key Cryptosystem Using Purely Cubic Fields, *J. Cryptology*, vol. 11 (1998), 109-124.
17. V.SHOUP, A Proposal for an ISO Standard for Public Key Encryption (v. 2.1), ISO/IEC JTC1/SC27, N2563, 2001. Available at <http://shoup.net/papers/> or <http://eprint.iacr.org/>
18. T.TAKAGI, Fast RSA-type cryptosystem modulo $p^k q$, *Advances in Cryptology – CRYPTO ’98*, LNCS 1462, 1998, 318-326.
19. H.C.WILLIAMS, A modification of the RSA public key encryption procedure, *IEEE Trans. on Information Theory*, IT-26, 6 (1980), 726-729.
20. H.C.WILLIAMS, Some public-key crypto-functions as intractable as factorization, *Cryptologia*, vol. 9, no.3 (1985), 223-237.
21. H.C.WILLIAMS, An M^3 public-key encryption scheme, *Advances in Cryptology – CRYPTO’85 Proceedings*, Springer-Verlag (1986), 358-368.

A Proof of Theorem 1

We begin with the next lemma.

Lemma 5. *If we identify \mathbb{Z}_N with integers between 0 and $N-1$, then as integers, for $0 < i < g_p$ and $0 < j < g_q$, the followings hold.*

$$\gcd(\phi(\zeta_{p,g}^i, 1) - 1, N) = q, \quad \gcd(\phi(1, \zeta_{q,g}^j) - 1, N) = p^d.$$

Proof. Since $\phi(\zeta_{p,g}^i, 1) - 1 \bmod p^d = \zeta_{p,g}^i - 1 \neq 0$, and $\phi(\zeta_{p,g}^i, 1) - 1 \bmod q = 1 - 1 = 0$, we can see that the greatest common multiple of this and N is equal to q . Similarly we can prove the second equation.

Let us denote $H_{N,g}$ the set of roots of unity in Lemma 5:

$$H_{N,g} = \{\phi(\zeta_{p,g}^i, 1) \mid 0 < i < g_p\} \cup \{\phi(1, \zeta_{q,g}^j) \mid 0 < j < g_q\} \subset Z_{N,g}.$$

Moreover, let us define $G_{N,g}$ as follows:

$$G_{N,g} = \{x \in Z_{N,g} \mid x^e \in H_{N,g} \text{ for some divisor } e \text{ of } g\}.$$

From the definition, $H_{N,g} \subset G_{N,g}$. Then the number of elements in $G_{N,g}$, denoted by $g_{N,g}$, is given by following.

Lemma 6. $g_{N,g} = g_p g_q - \sum_{e \mid \gcd(g_p, g_q)} \varphi(e)^2$.

Proof. $\phi(\zeta_{p,g}^i, \zeta_{q,g}^j) \in Z_{N,g}$ is in $G_{N,g}$ if and only if the order of p -part of $\phi(\zeta_{p,g}^i, \zeta_{q,g}^j)$ is different from that of q -part of it. Hence, elements in $Z_{N,g} \setminus G_{N,g}$ are which have same order in p -part and q -part, in this case, the orders are divisors of $\gcd(g_p, g_q)$. For each divisor $e \mid \gcd(g_p, g_q)$, the number of elements which have the order of p and q -part e is equal to $\varphi(e)^2$, which gives the desired result.

Proof of Theorem 1: We now give the proof of Theorem 1. Under the assumptions, let us put $g_p = \gcd(g, p-1)$, $g_q = \gcd(g, q-1)$, then f is $g_p g_q : 1$ ($g_p g_q > 1$) function (Of course, the adversary does not know p, q , but she knows that f is not injective). We assume that there exists a PPT algorithm A which computes a preimage of f . That is, A has input $k, d, g, \mathcal{D}_1, \mathcal{D}_2, N, y$, and it outputs x' in $f^{-1}(y) = \{x \in \mathbb{Z}_N^* \mid x^g = y\}$ with non-negligible probability. Using A , we construct an algorithm M which factors N as follows:

Input of M : $k, d, g, \mathcal{D}_1, \mathcal{D}_2, N$

Output of M : a prime factor of N

1. Choose randomly $x \in \mathbb{Z}_N^*$ ($x \neq 1$).
2. Calculate $y = x^g \bmod N$.
3. Input $(k, d, g, \mathcal{D}_1, \mathcal{D}_2, N, y)$ to A .
4. For an output x' of A , if $y \neq x'^g \bmod N$, then **Fail**.
5. Calculate $z = x'/x \bmod N$.

6. For each divisor e of g , calculate $w = \gcd((z^e - 1 \bmod N), N)$, if w is non-trivial divisor of N , then output w and end, otherwise **Fail**.

In Step 6, it outputs non-trivial divisor w if and only if $z \in G_{N,g}$, hence the success probability in Step 6 is equal to $g_{N,g}/g_p g_q$. If we put the success probability of A (that is, the probability such that it does not **Fail** in Step 4) to $\text{Adv}(A) = \epsilon$, then, by Lemma 6, the final success probability of M , namely, the probability such that M factors N , is equal to

$$\frac{g_{N,g}}{g_p g_q} \cdot \epsilon = \frac{g_p g_q - \sum_{e|\gcd(g_p, g_q)} \varphi(e)^2}{g_p g_q} \cdot \epsilon,$$

which is non-negligible by the assumptions.

B Proofs of Lemmas in Section 2.3

Proof of Lemma 3: By the assumptions, we have $y_{i+1} - x_i^g \bmod p^i q = y_i - x_i^g \bmod p^i q = 0$. Hence $x_{i+1} \bmod p = x_i \bmod p = x_p$. Moreover, $x_{i+1}^g = x_i^g + g x_i^{g-1} \eta_y (y_{i+1} - x_i^g) \bmod p^{i+1} q$, and by the assumption on x_i and the definition of η_y , we have $g x_i^{g-1} \eta_y \bmod p = g x_p^{g-1} \eta_y \bmod p = 1 \bmod p$. Therefore, we have $x_{i+1}^g \bmod p^{i+1} q = x_i^g + y_{i+1} - x_i^g \bmod p^{i+1} q = y_{i+1}$.

Proof of Proposition 1: Let a be a generator of the cyclic group $(\mathbb{Z}_p^*)^g$. The order of a is equal to $(p-1)/g_0$. If there exists an integer α which satisfies the condition, we have $(a^\alpha)^g = a$, thus it must be $\alpha \cdot g \equiv 1 \pmod{(p-1)/g_0}$. That is, g must be prime to $(p-1)/g_0$. Conversely, if $\gcd(g, (p-1)/g_0) = 1$, then let α be as above, it is directly checked that it satisfies the condition. As the order of a is $(p-1)/g_0$, we have $a^{\alpha g} = a^{1+u\{(p-1)/g_0\}} = a \pmod{p}$. Moreover, $u = \{-(p-1)/g_0\}^{-1} \bmod g$, hence the numerator of α is divided by g , thus α is an integer. Since $u \leq g-1$, $g_0 \leq g < p-1$, we have $1 + u\{(p-1)/g_0\} \leq 1 + (g-1)\{(p-1)/g_0\} < g(p-1)/g_0 \leq g(p-1)$. Thus α satisfies $1 \leq \alpha < p-1$.

C Proofs of the Security of Proposed Schemes

Proof of Theorem 2: Let \mathcal{A} be a forger which (t, q_H, q_s, ϵ) -breaks the signature scheme $F\text{-FDH}$. The input of \mathcal{A} is a public key (N, a) . \mathcal{A} has oracle access to random oracle H . Then we construct the factoring algorithm I which can (t_I, ϵ_I) -break by using \mathcal{A} . The input of I is $N \in \mathcal{N}$. I gives \mathcal{A} the public key N (we assume that I and \mathcal{A} know a as a system parameter). After this, \mathcal{A} begins to make sign queries and hash queries. For these queries, I behaves as follows.

If \mathcal{A} makes a sign query without having made the corresponding hash query, I at once goes ahead and makes the hash query itself, and then corresponds for sign query as described below. Similarly for the output forgery, thus we may assume that if \mathcal{A} makes a sign query or outputs a forgery, then it has already made corresponding hash query. Hence, effective number of hash queries is at most $q(k) = q_H(k) + q_s(k) + 1$.

To answer queries, I makes the query-mapping table (Q, A) as follows: Start with $Q = A = \phi$ (empty set). Suppose \mathcal{A} makes a hash query m .

If $m \notin Q$, then I chooses randomly $r \in_R \mathbb{Z}_N^*$ and $i \in \{0, 1, 2\}$, returns $H(m) = r^3/a^i \bmod N$, and sets $Q = Q \cup \{m\}$, $A = A \cup \{(m, r, i, H(m))\}$.

If $m \in Q$, then I finds corresponding $(m, r, i, H(m)) \in A$ and returns $H(m) (= r^3/a^i \bmod N)$.

Next, suppose that \mathcal{A} makes a sign query m . As mentioned above, we can assume that there was already a hash query m , hence there exists corresponding $(m, r, i, H(m)) \in A$. I finds this and returns r as the signature for m .

Finally, suppose that \mathcal{A} outputs a forgery (\hat{m}, \hat{s}) . If \hat{s} is valid, then for some i , $a^i H(\hat{m}) = \hat{s}^3 \bmod N$. N is chosen randomly and H is random from our construction of I . Hence \mathcal{A} can not distinguish the behavior of I from the original game, thus \mathcal{A} succeeds this simulation with original success probability ϵ .

On the other hand, by the assumption, $\hat{m} \in Q$, thus there exists the corresponding $(\hat{m}, \hat{r}, i, H(\hat{m})) \in A$ and it holds $\hat{r}^3 = \hat{s}^3 \bmod N$. Suppose that \hat{s} is valid. From the argument in Theorem 1, using the above equations (if $\hat{r} \neq \hat{s}$), a non-trivial factor of N can be calculated with success probability $2/3$. Thus I succeeds in factoring N with probability $\epsilon_I = (2/3)\epsilon$.

Let $t_0(k)$ be processing time for a modular multiplication with k -bit modulus. I carries out 3-modular multiplications for each hash query, hence also from Theorem 1, the processing time t' of I is given by

$$\begin{aligned} t' &\leq t + O(k^3) + 3(q_H + q_s + 1)t_0(k) \\ &= t + (q_H + q_s + 2)O(k^3). \end{aligned}$$

Proof of Theorem 3, 4: Theorem 3 can be proven just like Theorem 2 except for the behavior of simulator I .

First, I makes the query mapping table (Q_H, A_H) , (Q_G, A_G) starting with empty sets. Suppose the forger \mathcal{A} makes a H -query m . If $m \notin Q_H$, then I chooses $r \in_R \mathbb{Z}_N^*$ and $i \in \{0, 1, 2\}$ calculate $y = w_1 || w_2 = r^3/a^i \bmod N$ ($|w_1| = k_1$, $|w_2| = k_2$) and sets $Q_H = Q_H \cup \{m\}$, $A_H = A_H \cup \{(m, r, i, w_1, w_2)\}$, $Q_G = Q_G \cup \{w_1\}$, $A_G = A_G \cup \{(w_1, w_2)\}$. Finally, I returns $H(m) = w_1$. If $m \in Q_H$, then I finds corresponding $(m, r, i, w_1, w_2) \in A_H$ and returns $H(m) = w_1$.

When \mathcal{A} makes a G -query w_1 , if $w_1 \in Q_G$, then I finds corresponding $(w_1, w_2) \in A_G$ and returns $G(w_1) = w_2$, else I generates randomly r_2 , $|r_2| = k_2$, sets $Q_G = Q_G \cup \{w_1\}$, $A_G = A_G \cup \{(w_1, r_2)\}$ and returns $G(w_1) = r_2$.

Finally, suppose that \mathcal{A} makes a sign query m . We can assume that $m \in Q_H$, hence I can find corresponding $(m, r, i, w_1, w_2) \in A_H$ and returns r as a signature for m .

Then, as in Theorem 2, we can see that I can factor the modulus using the forgery outputted by \mathcal{A} with the indicated probability and processing time.

The proof for Theorem 4 is similar as in Theorem 3, so we omit the detail.

D Validity of the Algorithm Φ

We can easily obtain the algorithm Φ from Corollary 1 and the following lemma.

Let $a \in \mathbb{Z}_q^*$ be a non-cubic residue and fix. In each case $q \bmod 9 = 4$ or 7 , define the followings: in case of $q \bmod 9 = 4$, $\alpha = (2q+1)/9$, $\zeta = a^{(q-1)/3} \bmod q$, and in case of $q \bmod 9 = 7$, $\alpha = (q+2)/9$, $\zeta = a^{(2(q-1))/3} \bmod q$. Moreover, put $\beta = \alpha - 1$ and $b = a^\alpha \bmod q$.

Lemma 7. *For $w \in \mathbb{Z}_q^*$, put $w_1 = w^\beta \bmod q$, $w_2 = w_1 \cdot w \bmod q$ and $w_3 = w_1 \cdot w_2^2 \bmod q$. Then $w_3 \in \{1, \zeta, \zeta^2\}$, and we have followings: $w_3^2 \equiv w \bmod q$ if $w_3 = 1$, $(bw_2)^3 \equiv aw \bmod q$ if $w_3 = \zeta$ and $(b^2w_2)^3 \equiv a^2w \bmod q$ otherwise ($w_3 = \zeta^2$).*

Proof. By the assumptions and Lemma 1, ζ is a non-trivial cubic root of unity. Note that $\chi_{q,3}(a) = \zeta$ ($q \bmod 9 = 4$), $= \zeta^2$ ($q \bmod 9 = 7$). Moreover, in case of $q \bmod 9 = 4$, $w_3 = w^{2(q-4)/9+2(2q+1)/9} = w^{2(q-1)/3} = \chi_{q,3}(w)^2$, in case of $q \bmod 9 = 7$, $w_3 = w^{(q-7)/9+2(q+2)/9} = w^{(q-1)/3} = \chi_{q,3}(w)$. Hence, by Lemma 1, we have $w_3 \in \{1, \zeta, \zeta^2\}$, and $w_3 = 1$ means that w is a cubic residue. In this case, by Lemma 1, $w_2 = w^\alpha$ is a cubic root of w . In case of $w_3 = \zeta$, if $q \bmod 9 = 4$, then, by the above, we have $\chi_{q,3}(w) = \zeta^2$, and $\chi_{q,3}(aw) = \zeta \cdot \zeta^2 = 1$, moreover since $bw_2 = (aw)^\alpha \bmod q$, by Lemma 1, we have the result. In case of $q \bmod 9 = 7$ or $w_3 = \zeta^2$, it can be shown similarly.

E Decryption of RSA Function

We briefly recall that the decryption algorithm for Multi-Prime RSA [13].

Let p, q, r be distinct prime numbers, $N = pqr$, $z_q = p^{-1} \bmod q$, and $z_r = (pq)^{-1} \bmod r$. Let $1 < d < N$ be an integer such that $\gcd(d, (p-1)(q-1)(r-1)) = 1$ and $d_p = d \bmod p$, $d_q = d \bmod q$, $d_r = d \bmod r$. In the case of p, q, r are known, for any $C \in \mathbb{Z}_N^*$, we can calculate $M = C^d \bmod M$ (this is the preimage of C by the RSA function $x^e \bmod N$, where $e = d^{-1} \bmod (p-1)(q-1)(r-1)$) as follows:

Step 1. $C_p = C \bmod p$, $C_q = C \bmod q$, $C_r = C \bmod r$.

Step 2. $M_p = C_p^{d_p} \bmod p$, $M_q = C_q^{d_q} \bmod q$, $M_r = C_r^{d_r} \bmod r$.

Step 3. $M_{pq} = M_p + p((M_q - M_p)z_q \bmod q)$.

Step 4. $M = M_{pq} + (pq)((M_r - M_{pq})z_r \bmod r)$.

Step 5. Output M and end.

Certificateless Public Key Cryptography

Sattam S. Al-Riyami and Kenneth G. Paterson*

Information Security Group

Royal Holloway, University of London, Egham, Surrey, TW20 0EX

{s.al-riyami,kenny.paterson}@rhul.ac.uk

Abstract. This paper introduces and makes concrete the concept of *certificateless public key cryptography* (CL-PKC), a model for the use of public key cryptography which avoids the inherent escrow of identity-based cryptography and yet which does not require certificates to guarantee the authenticity of public keys. The lack of certificates and the presence of an adversary who has access to a master key necessitates the careful development of a new security model. We focus on certificateless public key encryption (CL-PKE), showing that a concrete pairing-based CL-PKE scheme is secure provided that an underlying problem closely related to the Bilinear Diffie-Hellman Problem is hard.

1 Introduction

The main difficulty today in developing secure systems based on public key cryptography is not the problem of choosing appropriately secure algorithms or implementing those algorithms. Rather, it is the deployment and management of infrastructures to support the authenticity of cryptographic keys: there is a need to provide an assurance to the user about the relationship between a public key and the identity (or authority) of the holder of the corresponding private key. In a traditional Public Key Infrastructure (PKI), this assurance is delivered in the form of certificate, essentially a signature by a Certification Authority (CA) on a public key [1]. The problems of PKI technology are well documented, see for example [16]. Of note are the issues associated with certificate management, including revocation, storage and distribution and the computational cost of certificate verification. These are particularly acute in processor or bandwidth-limited environments [9].

Identity-based public key cryptography (ID-PKC), first proposed by Shamir [22], tackles the problem of authenticity of keys in a different way to traditional PKI. In ID-PKC, an entity's public key is derived directly from certain aspects of its identity. Private keys are generated for entities by a trusted third party called a private key generator (PKG). The first fully practical and secure identity-based public key encryption scheme was presented in [5]. Since then, a rapid development of ID-PKC has taken place, see [18] for a brief survey. It has also been illustrated in [8,18,24] how ID-PKC can be used as a tool to enforce what

* This author supported by the Nuffield Foundation, NUF-NAL 02.

might be termed “cryptographic work-flows”, that is, sequences of operations (e.g. authentications) that need to be performed by an entity in order to achieve a certain goal.

The direct derivation of public keys in ID-PKC eliminates the need for certificates and some of the problems associated with them. On the other hand, the dependence on a PKG who uses a system-wide master key to generate private keys inevitably introduces key escrow to ID-PKC systems. For example, the PKG can decrypt any ciphertext in an identity-based public key encryption scheme. Equally problematical, the PKG could forge any entity’s signatures in an identity-based signature scheme, so ID-PKC cannot offer true non-repudiation in the way that traditional PKI can. The escrow problem can be solved to a certain extent by the introduction of multiple PKGs and the use of threshold techniques, but this necessarily involves extra communication and infrastructure. Moreover, the compromise of the PKG’s master key could be disastrous in an ID-PKC system, and usually more severe than the compromise of a CA’s signing key in a traditional PKI. For these reasons, it seems that the use of ID-PKC may be restricted to small, closed groups or to applications with limited security requirements.

1.1 Certificateless Public Key Cryptography

In this paper, we introduce a new paradigm for public key cryptography, which we name certificateless public key cryptography (CL-PKC). Our concept grew out of a search for public key schemes that do not require the use of certificates and yet do not have the built-in key escrow feature of ID-PKC. The solution we propose enjoys both of these properties; it is a model for the use of public key cryptography that is intermediate between traditional PKI and ID-PKC.

We demonstrate that our concept of CL-PKC can be made real by specifying certificateless encryption and signature schemes. We prove that the encryption scheme is secure in a new and appropriate model, given the hardness of an underlying computational problem. Further development of our concept and more certificateless schemes can be found in the full version of this paper, [2].

1.2 Defining CL-PKC

We sketch the defining characteristics of CL-PKC.

A CL-PKC system still makes use of TTP which we name the key generating centre (KGC). By way of contrast to the PKG in ID-PKC, this KGC does *not* have access to entities’ private keys. Instead, the KGC supplies an entity A with a *partial private key* D_A which the KGC computes from an identifier ID_A for the entity and a master key. Note that we will often equate A with its identifier ID_A . The process of supplying partial private keys should take place confidentially and authentically: the KGC must ensure that the partial private keys are delivered securely to the correct entities. Identifiers can be arbitrary strings.

The entity A then combines its partial private key D_A with some secret information to generate its actual private key S_A . In this way, A ’s private key is

not available to the KGC. The entity A also combines its secret information with the KGC's public parameters to compute its public key P_A . Note that A need not be in possession of S_A before generating P_A : all that is needed to generate both is the same secret information. The system is not identity-based, because the public key is no longer computable from an identity (or identifier) alone.

Entity A 's public key might be made available to other entities by transmitting it along with messages (for example, in a signing application) or by placing it in a public directory (this would be more appropriate for an encryption setting). But no further security is applied to the protection of A 's public key. In particular, there is no certificate for A 's key. To encrypt a message to A or verify a signature from A , entity B makes use of P_A and ID_A .

A more formal model for certificateless public key encryption (CL-PKE) will be given in Section 3. Much of this model is also applicable for our other certificateless primitives.

1.3 An Adversarial Model for CL-PKC

Because of the lack of authenticating information for public keys (in the form of a certificate, for example), we must assume that an adversary can replace A 's public key by a false key of its choice. This might seem to give the adversary tremendous power and to be disastrous for CL-PKC. However, we will see that an active adversary who attacks our concrete schemes in this way gains nothing useful: without the correct private key, whose production requires the partial private key and therefore the cooperation of the KGC, an adversary will not be able to decrypt ciphertexts encrypted under the false public key, produce signatures that verify with the false public key, and so on.

Of course, we must assume that the KGC does not mount an attack of this type: armed with the partial private key and the ability to replace public keys, the KGC could impersonate any entity in generating a private/public key pair and then making the public key available. Thus we must assume that, while the KGC is in possession of the master key and hence all partial private keys, it is trusted not to replace entities' public keys. However, we assume that the KGC might engage in other adversarial activity, eavesdropping on ciphertexts and making decryption queries, for example. In this way, users invest roughly the same level of trust in the KGC as they would in a CA in a traditional PKI – it is rarely made explicit, but such a CA is always assumed not to issue new certificates binding arbitrary public keys and entity combinations of its choice, and especially not for those where it knows the corresponding private key! When compared to ID-PKC, the trust assumptions made of the trusted third party in CL-PKC are much reduced: in ID-PKC, users must trust the PKG not to abuse its knowledge of private keys in performing passive attacks, while in CL-PKC, users need only trust the KGC not to actively propagate false public keys.

The word *roughly* here merits further explanation. In a traditional PKI, if the CA forges certificates, then the CA can be identified as having misbehaved through the existence of two valid certificates for the same identity. This is not the case in our schemes: a new public key could have been created by the

legitimate user or by the KGC, and it cannot be easily decided which is the case. The terminology of [15] is useful here: our schemes achieve trust level 2, whereas a traditional PKI reaches trust level 3. However, we can further strengthen security against a malicious KGC in our schemes by allowing entities to bind together their public keys and identities. Now the existence of two different, working public keys for the same identity will identify the KGC as having misbehaved in issuing both corresponding partial private keys. Details of this modification can be found in Section 5.1. With this binding in place, our schemes do reach trust level 3.

In Section 3, we will present an adversarial model for CL-PKE which captures these capabilities in a formal way. The model we present there is a natural generalization of the fully adaptive, multi-user model of [5] to the CL-PKC setting, and involves two distinct types of adversary: one who can replace public keys at will and another who has knowledge of the master key but does not replace public keys. Given our detailed development of this model, the adaptations to existing models that are needed to produce adversarial models for other certificateless primitives become straightforward.

1.4 Implementation and Applications of CL-PKC

Our presentation of CL-PKC schemes will be at a fairly abstract level, in terms of bilinear maps on groups. However, the concrete realization of these schemes using pairings on elliptic curves is now becoming comparatively routine, after the work of [3,6,7,12] on implementation of pairings and selection of curves with suitable properties. All the schemes we present use a small number of pairing calculations for each cryptographic operation, and some of these can usually be eliminated when repeated operations involving the same identities take place. Public and private keys are small in size: two elliptic curve points for the public key and one for the private key.

The infrastructure needed to support CL-PKC is lightweight when compared to a traditional PKI. This is because, just as with ID-PKC, the need to manage certificates is completely eliminated. This immediately makes CL-PKC attractive for low-bandwidth, low-power situations. However, it should be pointed out that recently introduced signatures schemes enjoying very short signatures [7] could be used to significantly decrease the size of certificates and create a lightweight PKI. Our CL-PKC signature scheme can also support true non-repudiation, because private keys remain in the sole possession of their legitimate owners.

Revocation of keys in CL-PKC systems can be handled in the same way as in ID-PKC systems. In [5] the idea of appending validity periods to identifiers ID_A is given as one convenient solution. In the context of CL-PKC, this ensures that any partial private key, and hence any private key, has a limited shelf-life.

As will become apparent, our CL-PKC schemes are actually very closely related to existing pairing-based ID-PKC schemes. One consequence of this is that any infrastructure deployed to support pairing-based ID-PKC (e.g. a PKG) can also be used to support our CL-PKC schemes too: in short, the two types of scheme can peacefully co-exist. In fact, an entity can be granted a private key for

a pairing-based ID-PKC scheme and immediately convert it into a private key for our CL-PKC scheme. In this way, an entity who wishes to prevent the PKG exploiting the escrow property of an identity-based system can do so, though at the cost of losing the identity-based nature of its public key.

Although our CL-PKC schemes are no longer identity-based, they do enjoy the property that an entity's private key can be determined after its public key has been generated and used. This is a useful feature. An entity B can encrypt a message for A using A 's chosen public key and an identifier ID_A of B 's choice. This identifier should contain A 's identity but might also contain a condition that A must demonstrate that it satisfies before the KGC will deliver the corresponding partial private key (which in turn allows A to compute the right private key for decryption). For more applications of "cryptographic workflows" which cannot be supported using certificate-based systems, see [18,24].

1.5 Related Work

Our work on CL-PKC owes much to the pioneering work of Boneh and Franklin [5,6] on identity-based public key encryption. In fact, our CL-PKE scheme is derived from the scheme of [5] by making a very simple modification (albeit, one with far-reaching consequences). Our security proofs require significant changes and new ideas to handle our new types of adversary. Likewise, our signature and other schemes [2] also arise by adapting existing ID-PKC schemes. Another alternative to traditional certificate-based PKI called self-certified keys was introduced by Girault [15] and further developed in [19,21]. The properties of the schemes presented in [15,19,21] are compared to CL-PKC in the full version [2].

Recent and independent work of Gentry [13] simplifies certificate management in traditional PKI systems in a very neat way by exploiting pairings. Gentry's scheme is presented in the context of a traditional PKI model, whereas our work departs from the traditional PKI and ID-PKC models to present a new paradigm for the use of public-key cryptography. Moreover, the concrete realizations of the two models are different. However, it is possible to re-cast Gentry's work to divorce it from the setting of a traditional PKI. Further discussion can be found in [2].

2 Background Definitions

Throughout the paper, \mathbb{G}_1 denotes an additive group of prime order q and \mathbb{G}_2 a multiplicative group of the same order. We let P denote a generator of \mathbb{G}_1 . For us, a pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties: (1) The map e is bilinear: given $Q, W, Z \in \mathbb{G}_1$, we have $e(Q, W + Z) = e(Q, W) \cdot e(Q, Z)$ and $e(Q + W, Z) = e(Q, Z) \cdot e(W, Z)$. (2) The map e is non-degenerate: $e(P, P) \neq 1_{\mathbb{G}_2}$. (3) The map e is efficiently computable.

Typically, the map e will be derived from either the Weil or Tate pairing on an elliptic curve over a finite field. We refer to [3,6,7,12] for a more comprehensive description of how these groups, pairings and other parameters should be selected in practice for efficiency and security.

We also introduce here the computational problems that will form the basis of security for our CL-PKC schemes.

Bilinear Diffie-Hellman Problem (BDHP): Let $\mathbb{G}_1, \mathbb{G}_2, P$ and e be as above. The BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is as follows: Given $\langle P, aP, bP, cP \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$, compute $e(P, P)^{abc} \in \mathbb{G}_2$. An algorithm \mathcal{A} has advantage ϵ in solving the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ if $\Pr [\mathcal{A}(\langle P, aP, bP, cP \rangle) = e(P, P)^{abc}] = \epsilon$.

Here the probability is measured over the random choices of $a, b, c \in \mathbb{Z}_q^*$ and the random bits of \mathcal{A} .

Generalized Bilinear Diffie-Hellman Problem (GBDHP): Let $\mathbb{G}_1, \mathbb{G}_2, P$ and e be as above. The GBDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is as follows: Given $\langle P, aP, bP, cP \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$, output a pair $\langle Q \in \mathbb{G}_1^*, e(P, Q)^{abc} \in \mathbb{G}_2 \rangle$. An algorithm \mathcal{A} has advantage ϵ in solving the GBDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ if $\Pr [\mathcal{A}(\langle P, aP, bP, cP \rangle) = \langle Q, e(P, Q)^{abc} \rangle] = \epsilon$.

Here the probability is measured over the random choices of $a, b, c \in \mathbb{Z}_q^*$ and the random bits of \mathcal{A} .

Notice that the BDHP is a special case of the GBDHP in which the algorithm outputs the choice $Q = P$. While the GBDHP may appear to be in general easier to solve than the BDHP because the solver gets to choose Q , we know of no polynomial-time algorithm for solving either when the groups $\mathbb{G}_1, \mathbb{G}_2$ and pairing e are appropriately selected. If the solver knows $s \in \mathbb{Z}_q^*$ such that $Q = sP$, then the problems are of course equivalent.

BDH Parameter Generator: As in [5], the formal output of this randomized algorithm is a triple $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ where \mathbb{G}_1 and \mathbb{G}_2 are of prime order q and $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a pairing.

Our security proofs will yield reductions to the BDHP or GBDHP in groups generated by a BDH parameter generator \mathcal{IG} .

3 Certificateless Public Key Encryption

In this section we present a formal definition for a certificateless public key encryption (CL-PKE) scheme. We also examine the capabilities which may be possessed by the adversaries against such a scheme and give a security model for CL-PKE.

A CL-PKE scheme is specified by seven randomized algorithms.

Setup: This algorithm takes security parameter k and returns the system parameters **params** and **master-key**. The system parameters includes a description of the message space \mathcal{M} and ciphertext space \mathcal{C} . Usually, this algorithm is run by the KGC. We assume throughout that **params** are publicly and authentically available, but that only the KGC knows **master-key**.

Partial-Private-Key-Extract: This algorithm takes **params**, **master-key** and an identifier for entity A , $ID_A \in \{0, 1\}^*$, as input. It returns a partial private key D_A . Usually this algorithm is run by the KGC and its output is transported to entity A over a confidential and authentic channel.

Set-Secret-Value: This algorithm takes as inputs **params** and an entity A 's identifier ID_A as inputs and outputs A 's secret value x_A .

Set-Private-Key: This algorithm takes **params**, an entity A 's partial private key D_A and A 's secret value x_A as input. The value x_A is used to transform D_A into the (full) private key S_A . The algorithm returns S_A .

Set-Public-Key: This algorithm takes **params** and entity A 's secret value x_A as input and from these constructs the public key P_A for entity A .

Normally both **Set-Private-Key** and **Set-Public-Key** are run by an entity A for itself, after running **Set-Secret-Value**. The same secret value x_A is used in each. Separating them makes it clear that there is no need for a temporal ordering on the generation of public and private keys in our CL-PKE scheme. Usually, A is the only entity in possession of S_A and x_A , and x_A will be chosen at random from a suitable and large set.

Encrypt: This algorithm takes as inputs **params**, a message $M \in \mathcal{M}$, and the public key P_A and identifier ID_A of an entity A . It returns either a ciphertext $C \in \mathcal{C}$ or the null symbol \perp indicating an encryption failure. This will always occur in the event that P_A does not have the correct form. In our scheme, this is the only way an encryption failure will occur.

Decrypt: This algorithm takes as inputs **params**, $C \in \mathcal{C}$, and a private key S_A . It returns a message $M \in \mathcal{M}$ or a message \perp indicating a decryption failure.

Naturally, we insist that output M should result from applying algorithm **Decrypt** with inputs **params**, S_A on a ciphertext C generated by using algorithm **Encrypt** with inputs **params**, P_A , ID_A on message M .

3.1 Security Model for CL-PKE

Given this formal definition of a CL-PKE scheme, we are now in a position to define adversaries for such a scheme. The standard definition for security for a public key encryption scheme involves indistinguishability of encryptions against a fully-adaptive chosen ciphertext attacker (IND-CCA) [4,10,20]. In this definition, there are two parties, the adversary \mathcal{A} and the challenger \mathcal{C} . The adversary operates in three phases after being presented with a random public key. In Phase 1, \mathcal{A} may make decryption queries on ciphertexts of its choice. In the Challenge Phase, \mathcal{A} chooses two messages M_0, M_1 and is given a challenge ciphertext C^* for one of these two messages M_b by the challenger. In Phase 2, \mathcal{A} may make further decryption queries, but may not ask for the decryption of C^* . The attack ends with \mathcal{A} 's guess b' for the bit b . The adversary's advantage is defined to be $\text{Adv}(\mathcal{A}) = 2(\Pr[b' = b] - \frac{1}{2})$.

This model was strengthened for ID-PKC in [5] to handle adversaries who can extract the private keys of arbitrary entities and who choose the identity ID_{ch} of the entity on whose public key they are challenged. This extension is appropriate because the compromise of some entities' private keys should not affect the security of an uncompromised entity's encryptions.

Here, we extend the model of [5] to allow adversaries who can extract partial private keys, or private keys, or both, for identities of their choice. Given that

our scheme has no certificates, we must further strengthen the model to allow for adversaries who can replace the public key of any entity with a value of their choice. We must also consider carefully how a challenger should respond to key extraction and decryption queries for identities whose public keys have been changed.

Here then is a list of the actions that a general adversary against a CL-PKE scheme may carry out and a discussion of each action.

- (1) **Extract partial private key of A :** \mathcal{C} responds by running algorithm **Partial-Private-Key-Extract** to generate the partial private key D_A for entity A .
- (2) **Extract private key for A :** As in [5], we allow our adversary \mathcal{A} to make requests for entities' private keys. If A 's public key has not been replaced then \mathcal{C} can respond by running algorithm **Set-Private-Key** to generate the private key S_A for entity A (first running **Set-Secret-Value** for A if necessary). But it is unreasonable to expect \mathcal{C} to be able to respond to such a query if \mathcal{A} has already replaced A 's public key. Of course, we insist that \mathcal{A} does not at any point extract the private key for the selected challenge identity ID_{ch} .
- (3) **Request public key of A :** Naturally, we assume that public keys are available to \mathcal{A} . On receiving a first request for A 's public key, \mathcal{C} responds by running algorithm **Set-Public-Key** to generate the public key P_A for entity A (first running **Set-Secret-Value** for A if necessary).
- (4) **Replace public key of A :** \mathcal{A} can repeatedly replace the public key P_A for any entity A with any value P'_A of its choice. In our concrete CL-PKE schemes, our public keys will have a certain structure that is used to test the validity of public keys before any encryption. We assume here that the adversary's choice P'_A is a valid public key; this assumption can be removed (and our schemes remain secure) at the cost of some additional complexity in our definitions. Note that in our schemes, any entity can easily create public keys that are valid. The current value of an entity's public key is used by \mathcal{C} in any computations (for example, preparing a challenge ciphertext) or responses to \mathcal{A} 's requests (for example, replying to a request for the public key). We insist that \mathcal{A} cannot both replace the public key for the challenge identity ID_{ch} before the challenge phase *and* extract the partial private key for ID_{ch} in some phase – this would enable \mathcal{A} to receive a challenge ciphertext under a public key for which it could compute the private key.
- (5) **Decryption query for ciphertext C and entity A :** If \mathcal{A} has not replaced the public key of entity A , then \mathcal{C} responds by running the algorithm **Set-Private-Key** to obtain the private key S_A , then running **Decrypt** on ciphertext C and private key S_A and returning the output to \mathcal{A} . However, if \mathcal{A} has already replaced the public key of A , then in following this approach, \mathcal{C} would (in general) not decrypt using a private key matching the current public key. However, we insist that \mathcal{C} properly decrypts ciphertexts even for entities whose public keys have been replaced (these decryptions will be handled using special purpose knowledge extractors in our security proofs). This results in a very powerful security model because decryption queries made under public keys

that have been changed are potentially far more useful to \mathcal{A} . Naturally, as in [5], we prohibit \mathcal{A} from ever making a decryption query on the challenge ciphertext C^* for the combination of identity ID_{ch} and public key P_{ch} that was used to encrypt M_b . However \mathcal{A} is, for example, allowed to replace the public key for ID_{ch} with a new value and then request a decryption of C^* , or to change another entity A 's public key to P_{ch} (or any other value) and then request the decryption of C^* for entity A .

We also want to consider adversaries who are equipped with **master-key**, in order to model security against an eavesdropping KGC. As discussed in Section 1, we do not allow such an adversary to replace public keys: in this respect, we invest in the KGC the same level of trust as we do in a CA in a traditional PKI. So we will distinguish between two adversary types, with slightly different capabilities:

CL-PKE Type I Adversary: Such an adversary \mathcal{A}_I does not have access to **master-key**. However, \mathcal{A}_I may request public keys and replace public keys with values of its choice, extract partial private and private keys and make decryption queries, all for identities of its choice. As discussed above, we make several natural restrictions on such a Type I adversary: (1) \mathcal{A}_I cannot extract the private key for ID_{ch} at any point. (2) \mathcal{A}_I cannot request the private key for any identity if the corresponding public key has already been replaced. (3) \mathcal{A}_I cannot both replace the public key for the challenge identity ID_{ch} before the challenge phase *and* extract the partial private key for ID_{ch} in some phase. (4) In Phase 2, \mathcal{A}_I cannot make a decryption query on the challenge ciphertext C^* for the combination of identity ID_{ch} and public key P_{ch} that was used to encrypt M_b .

CL-PKE Type II Adversary: Such an adversary \mathcal{A}_{II} does have access to **master-key**, but may not replace public keys of entities. Adversary \mathcal{A}_{II} can compute partial private keys for itself, given **master-key**. It can also request public keys, make private key extraction queries and decryption queries, both for identities of its choice. The restrictions on this type of adversary are: (1) \mathcal{A}_{II} cannot replace public keys at any point. (2) \mathcal{A}_{II} cannot extract the private key for ID_{ch} at any point. (3) In Phase 2, \mathcal{A}_{II} cannot make a decryption query on the challenge ciphertext C^* for the combination of identity ID_{ch} and public key P_{ch} that was used to encrypt M_b .

Chosen ciphertext security for CL-PKE: We say that a CL-PKE scheme is semantically secure against an adaptive chosen ciphertext attack (“IND-CCA secure”) if no polynomially bounded adversary \mathcal{A} of Type I or Type II has a non-negligible advantage against the challenger in the following game:

Setup: The challenger takes a security parameter k and runs the **Setup** algorithm. It gives \mathcal{A} the resulting system parameters **params**. If \mathcal{A} is of Type I, then the challenger keeps **master-key** to itself, otherwise, it gives **master-key** to \mathcal{A} .

Phase 1: \mathcal{A} issues a sequence of requests, each request being either a partial private key extraction, a private key extraction, a request for a public key, a replace public key command or a decryption query for a particular entity. These queries

may be asked adaptively, but are subject to the rules on adversary behaviour defined above.

Challenge Phase: Once \mathcal{A} decides that Phase 1 is over it outputs the challenge identity ID_{ch} and two equal length plaintexts $M_0, M_1 \in \mathcal{M}$. Again, the adversarial constraints given above apply. The challenger now picks a random bit $b \in \{0, 1\}$ and computes C^* , the encryption of M_b under the current public key P_{ch} for ID_{ch} . If the output of the encryption is \perp , then \mathcal{A} has immediately lost the game (it has replaced a public key with one not having the correct form). Otherwise, C^* is delivered to \mathcal{A} .

Phase 2: \mathcal{A} issues a second sequence of requests as in Phase 1, again subject to the rules on adversary behaviour above.

Guess: Finally, \mathcal{A} outputs a guess $b' \in \{0, 1\}$. The adversary wins the game if $b = b'$. We define \mathcal{A} 's advantage in this game to be $\text{Adv}(\mathcal{A}) := 2(\Pr[b = b'] - \frac{1}{2})$.

4 CL-PKE Schemes from Pairings

In this section, we describe a pair of CL-PKE schemes. Our first scheme, **BasicCL-PKE**, is analogous to the scheme **BasicIdent** of [5], and is included only to serve as a warm-up for our main scheme **FullCL-PKE**. The main scheme is in turn an analogue of the scheme **FullIdent** of [5] and is IND-CCA secure, assuming the hardness of the GBDHP. We prove this in Theorem 1.

4.1 A Basic CL-PKE Scheme

We describe the seven algorithms needed to define **BasicCL-PKE**. We let k be a security parameter given to the **Setup** algorithm and \mathcal{IG} a BDH parameter generator with input k .

Setup: This algorithm runs as follows:

- (1) Run \mathcal{IG} on input k to generate output $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ where \mathbb{G}_1 and \mathbb{G}_2 are groups of some prime order q and $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a pairing.
- (2) Choose an arbitrary generator $P \in \mathbb{G}_1$.
- (3) Select a **master-key** s uniformly at random from \mathbb{Z}_q^* and set $P_0 = sP$.
- (4) Choose cryptographic hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^n$. Here n will be the bit-length of plaintexts.

The system parameters are $\text{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2 \rangle$. The **master-key** is $s \in \mathbb{Z}_q^*$. The message space is $\mathcal{M} = \{0, 1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0, 1\}^n$.

Partial-Private-Key-Extract: This algorithm takes as input an identifier $\text{ID}_A \in \{0, 1\}^*$, and carries out the following steps to construct the partial private key for entity A with identifier ID_A :

- (1) Compute $Q_A = H_1(\text{ID}_A) \in \mathbb{G}_1^*$.
- (2) Output the partial private key $D_A = sQ_A \in \mathbb{G}_1^*$.

The reader will notice that the partial private key of entity A here is identical to that entity's private key in the schemes of [5]. Also notice that A can verify the

correctness of the **Partial-Private-Key-Extract** algorithm output by checking $e(D_A, P) = e(Q_A, P_0)$.

Set-Secret-Value: This algorithm takes as inputs **params** and an entity A 's identifier ID_A as inputs. It selects $x_A \in \mathbb{Z}_q^*$ at random and outputs x_A as A 's secret value.

Set-Private-Key: This algorithm takes as inputs **params**, an entity A 's partial private key D_A and A 's secret value $x_A \in \mathbb{Z}_q^*$. It transforms partial private key D_A to private key S_A by computing $S_A = x_A D_A = x_A s Q_A \in \mathbb{G}_1^*$.

Set-Public-Key: This algorithm takes **params** and entity A 's secret value $x_A \in \mathbb{Z}_q^*$ as inputs and constructs A 's public key as $P_A = \langle X_A, Y_A \rangle$, where $X_A = x_A P$ and $Y_A = x_A P_0 = x_A s P$.

Encrypt: To encrypt $M \in \mathcal{M}$ for entity A with identifier $ID_A \in \{0, 1\}^*$ and public key $P_A = \langle X_A, Y_A \rangle$, perform the following steps:

- (1) Check that $X_A, Y_A \in \mathbb{G}_1^*$ and that the equality $e(X_A, P_0) = e(Y_A, P)$ holds. If not, output \perp and abort encryption.
- (2) Compute $Q_A = H_1(ID_A) \in \mathbb{G}_1^*$.
- (3) Choose a random value $r \in \mathbb{Z}_q^*$.
- (4) Compute and output the ciphertext: $C = \langle rP, M \oplus H_2(e(Q_A, Y_A)^r) \rangle$.

Notice that this encryption operation is identical to the encryption algorithm in the scheme **BasicIdent** of [5], except for the check on the structure of the public key in step 1 and the use of Y_A in place of $P_0 = P_{pub}$ in step 4.

Decrypt: Suppose $C = \langle U, V \rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key S_A , compute and output: $V \oplus H_2(e(S_A, U))$.

Notice that if $\langle U = rP, V \rangle$ is the encryption of M for entity A with public key $P_A = \langle X_A, Y_A \rangle$, the decryption is the inverse of encryption.

Again, the similarity to the decryption operation of **BasicIdent** should be apparent.

We have presented this scheme to help the reader understand our **FullCL-PKE** scheme, and so we do not analyse its security in detail.

4.2 A Full CL-PKE Scheme

Now that we have described our basic CL-PKE scheme, we add chosen ciphertext security to it, adapting the Fujisaki-Okamoto padding technique [11]. The algorithms for **FullCL-PKE** are as follows:

Setup: Identical to **Setup** for **BasicCL-PKE**, except that we choose two additional cryptographic hash functions $H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}_q^*$ and $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$.

The system parameters are **params** = $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2, H_3, H_4 \rangle$. The **master-key** and message space \mathcal{M} are the same as in **BasicCL-PKE**. The ciphertext space is now $\mathcal{C} = \mathbb{G}_1 \times \{0, 1\}^{2n}$.

Partial-Private-Key-Extract, **Set-Secret-Value**, **Set-Private-Key**, and **Set-Public-Key:** Identical to **BasicCL-PKE**.

Encrypt: To encrypt $M \in \mathcal{M}$ for entity A with identifier $\text{ID}_A \in \{0, 1\}^*$ and public key $P_A = \langle X_A, Y_A \rangle$, perform the following steps:

- (1) Check that $X_A, Y_A \in \mathbb{G}_1^*$ and that the equality $e(X_A, P_0) = e(Y_A, P)$ holds. If not, output \perp and abort encryption.
- (2) Compute $Q_A = H_1(\text{ID}) \in \mathbb{G}_1^*$.
- (3) Choose a random $\sigma \in \{0, 1\}^n$.
- (4) Set $r = H_3(\sigma, M)$.
- (5) Compute and output: $C = \langle rP, \sigma \oplus H_2(e(Q_A, Y_A)^r), M \oplus H_4(\sigma) \rangle$.

Decrypt: Suppose the ciphertext $C = \langle U, V, W \rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key S_A :

- (1) Compute $V \oplus H_2(e(S_A, U)) = \sigma'$.
- (2) Compute $W \oplus H_4(\sigma') = M'$.
- (3) Set $r' = H_3(\sigma', M')$ and test if $U = r'P$. If not, output \perp and reject C .
- (4) Output M' as the decryption of C .

When C is a valid encryption of M using P_A and ID_A , it is easy to see that decrypting C will result in an output $M' = M$. We note that W can be replaced by $W = E_{H_4(\sigma)}(M)$ where E denotes a semantically secure symmetric key encryption scheme as in [11] (though our security proofs will require some modifications to handle this case). This concludes the description of FullCL-PKE.

4.3 Security of the Scheme FullCL-PKE

We have the following theorem about the security of FullCL-PKE.

Theorem 1. *Let hash functions H_1, H_2, H_3 and H_4 be random oracles. Suppose further that there is no polynomially bounded algorithm that can solve the GB-DHP in groups generated by \mathcal{IG} with non-negligible advantage. Then FullCL-PKE is IND-CCA secure.*

This theorem follows from a sequence of lemmas that are proved in the appendices. It can be made into a concrete security reduction relating the advantage ϵ of a Type I or Type II attacker against FullCL-PKE to that of an algorithm to solve GBDHP or BDHP.

5 Further CL-PKC Schemes

In this section, we sketch another CL-PKC primitives: a signature scheme based on the identity-based scheme of [17]. We begin by outlining an alternative key generation technique which enhances the resilience of our schemes against a cheating KGC and allows for non-repudation of certificateless signatures.

5.1 An Alternative Key Generation Technique

Up to this point, we have assumed that the KGC is trusted to not replace the public keys of users and to only issue one copy of each partial private key, to the

correct recipient. This may involve an unacceptable level of trust in the KGC for some users. Our current set up also allows users to create more than one public key for the same partial private key. This can be desirable in some applications, but undesirable in others.

Here we sketch a simple binding technique which ensures that users can only create one public key for which they know the corresponding private key. In our technique, an entity A must first fix its secret value x_A and its public key $P_A = \langle X_A, Y_A \rangle$. We then re-define Q_A to be $Q_A = H_1(\text{ID}_A \| P_A)$ – now Q_A binds A 's identifier and public key. The partial private key delivered to entity A is still $D_A = sQ_A$ and the private key created by A is still xsQ_A . However, these are also now bound to A 's choice of public key. This binding effectively restricts A to using a single public key, since A can now only compute one private key from D_A .

This technique has a very important additional benefit: it reduces the degree of trust that users need to have in the KGC in our certificateless schemes. In short, the technique raises our schemes to trust level 3 in the trust hierarchy of [15], the same level as is enjoyed in a traditional PKI. Now, with our binding technique in place, a KGC who replaces an entity's public key will be implicated in the event of a dispute: the existence of two working public keys for an identity can only result from the existence of two partial private keys binding that identity to two different public keys; only the KGC could have created these two partial private keys. Thus our binding technique makes the KGC's replacement of a public key apparent and equivalent to a CA forging a certificate in a traditional PKI.

Theorem 1 still applies for our CL-PKE scheme with this binding in place because of the way in which H_1 is modelled as a random oracle. Notice too that with this binding in place, there is no longer any need to keep partial private keys secret: informally, knowledge of the key $D_A = sQ_A$ does not help an adversary to create the unique private key $S_A = xsQ_A$ that matches the particular public key P_A that is bound to D_A . When applied to the certificateless signature primitive in this section, the binding technique ensures a stronger form of non-repudiation: without the binding, an entity could always attempt to repudiate a signature by producing a second working public key and claiming that the KGC had created the signature using the first public key.

Even with this binding in place, the security analysis of our original encryption scheme (in which an adversary can replace public keys) is still important: it models the scenario where an adversary *temporarily* replaces the public key P_A of an entity A with a new value P'_A in an attempt to obtain a ciphertext which he can distinguish, and then resets the public key. In this case, our proof shows that the adversary does not gain any advantage in a distinguishing game unless he has access to the matching partial private key $D'_A = sH_1(\text{ID}_A \| P'_A)$. In turn, this partial private key should not be made available by the KGC. Of course, nothing can prevent a KGC from mounting an attack of this type, but the same applies for the CA in a traditional PKI.

5.2 A Certificateless Signature Scheme

We will describe a certificateless public-key signature (CL-PKS) scheme that is based on a provably secure ID-PKC signature scheme of [17].

In general, a CL-PKS scheme can be specified by seven algorithms: **Setup**, **Partial-Private-Key-Extract**, **Set-Secret-Value**, **Set-Private-Key**, **Set-Public-Key**, **Sign** and **Verify**. These are similar to the algorithms used to define a CL-PKE scheme: **Setup** and **params** are modified to include a description of the signature space \mathcal{S} , **Partial-Private-Key-Extract**, **Set-Secret-Value**, **Set-Private-Key** and **Set-Public-Key** are just as before and **Sign** and **Verify** are as follows:

Sign: This algorithm takes as inputs **params**, a message $M \in \mathcal{M}$ to be signed and a private key S_A . It outputs a signature $Sig \in \mathcal{S}$.

Verify: This algorithm takes as inputs **params**, a message $M \in \mathcal{M}$, the identifier ID_A and public key P_A of an entity A , and $Sig \in \mathcal{S}$ as the signature to be verified. It outputs **valid**, **invalid** or \perp .

Given this general description, we now outline a CL-PKS scheme:

Setup: This is identical to **Setup** for our scheme **BasicCL-PKE**, except that now there is only one hash function $H : \{0, 1\}^* \times \mathbb{G}_2 \rightarrow \mathbb{Z}_q^*$ and **params** is $\langle \mathbb{G}_1, \mathbb{G}_2, n, e, P, P_0, H \rangle$. The signature space is defined as $\mathcal{S} = \mathbb{G}_1 \times \mathbb{Z}_q^*$.

Partial-Private-Key-Extract, **Set-Secret-Value**, **Set-Private-Key** and **Set-Public-Key**: Identical to **BasicCL-PKE**.

Sign: To sign $M \in \mathcal{M}$ using the private key S_A , perform the following steps: (1) Choose random $a \in \mathbb{Z}_q^*$. (2) Compute $r = e(P, P)^a \in \mathbb{G}_2$. (3) Set $v = H(M, r) \in \mathbb{Z}_q^*$. (4) Compute $U = vS_A + aP \in \mathbb{G}_1$. (5) Output as the signature $\langle U, v \rangle$.

Verify: To verify a purported signature $\langle U, v \rangle$ on a message $M \in \mathcal{M}$ for identity ID_A and public key $\langle X_A, Y_A \rangle$: (1) Check that the equality $e(X_A, P_0) = e(Y_A, P)$ holds. If not, output \perp and abort verification. (2) Compute $r = e(U, P) \cdot e(Q_A, -Y_A)^v$. (3) Check if $v = H(M, r)$ holds. If it does, output **valid**, otherwise output **invalid**.

5.3 Other Schemes

The hierarchical encryption and signature schemes of [14] and the key agreement scheme of [23] can be adapted to our certificateless setting. These adaptations are presented in the full paper [2].

6 Conclusions

In this paper we introduced the concept of *certificateless public key cryptography*, a model for the use of public key cryptography that is intermediate between the identity-based approach and traditional PKI. We showed how our concept can be realized by specifying a certificateless public key encryption (CL-PKE) scheme that is based on bilinear maps. We showed that our CL-PKE scheme is secure in an appropriate model, assuming that the Generalized Bilinear Diffie-Hellman Problem (GBDHP) is hard. We also rounded out our treatment by briefly presenting a certificateless signature scheme.

Acknowledgement

We would like to thank Dan Boneh, Alex Dent, Steven Galbraith and Craig Gentry for their comments and helpful discussions on the paper.

References

1. C. Adams and S. Lloyd. *Understanding Public-Key Infrastructure – Concepts, Standards, and Deployment Considerations*. Macmillan, Indianapolis, USA, 1999.
2. S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. Cryptology ePrint Archive, Report 2003/126, 2003. <http://eprint.iacr.org/>.
3. P.S.L.M. Barreto *et al.* Efficient algorithms for pairing-based cryptosystems. In *Proc. CRYPTO 2002*, LNCS vol. 2442, pp. 354–368. Springer, 2002.
4. M. Bellare *et al.* Relations among notions of security for public-key encryption schemes. In *Proc. CRYPTO 1998*, LNCS vol. 1462. Springer, 1998.
5. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Proc. CRYPTO 2001*, LNCS vol. 2139, pp. 213–229. Springer, 2001.
6. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003.
7. D. Boneh, H. Shacham, and B. Lynn. Short signatures from the Weil pairing. In C. Boyd, editor, *Proc. ASIACRYPT 2001*, LNCS vol. 2248, pp. 514–532. Springer, 2001.
8. L. Chen *et al.* Certification of public keys within an identity based system. In A. H. Chan and V. D. Gligor, editors, *Information Security, 5th International Conference, ISC*, LNCS vol. 2433, pp. 322–333. Springer, 2002.
9. J. Dankers *et al.* Public key infrastructure in mobile systems. *IEE Electronics and Communcation Engineering Journal*, 14(5):180–190, 2002.
10. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM Journal of Computing*, 30(2):391–437, 2000.
11. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *Proc. CRYPTO 1999*, LNCS vol. 1666, pp. 537–554. Springer, 1999.
12. S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory 5th International Symposium, ANTS-V*, LNCS vol. 2369, pp. 324–337. Springer, 2002.
13. C. Gentry. Certificate-based encryption and the certificate revocation problem. In E. Biham, editor, *Proc. EUROCRYPT 2003*, LNCS vol. 2656, pp. 272–293. Springer, 2003.
14. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In Y. Zheng, editor, *Proc. ASIACRYPT 2002*, LNCS vol. 2501, pp. 548–566. Springer, 2002.
15. M. Girault. Self-certified public keys. In D. W. Davies, editor, *Proc. EUROCRYPT 1991*, LNCS vol. 547, pp. 490–497. Springer, 1992.
16. P. Gutmann. PKI: It’s not dead, just resting. *IEEE Computer*, 35(8):41–49, 2002.
17. F. Hess. Efficient identity based signature schemes based on pairings. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography 9th Annual International Workshop, SAC 2002*, LNCS vol. 2595, pp. 310–324. Springer, 2003.
18. K.G. Paterson. Cryptography from pairings: a snapshot of current research. *Information Security Technical Report*, 7(3):41–54, 2002.

19. H. Petersen and P. Horster. Self-certified keys – concepts and applications. In *3rd Int. Conference on Communications and Multimedia Security*. Chapman and Hall, 1997.
20. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attacks. In *Proc. CRYPTO 1991*, LNCS vol. 576, pp. 433–444. Springer, 1991.
21. S. Saeednia. Identity-based and self-certified key-exchange protocols. In V. Varadharajan, J. Pieprzyk, and Y. Mu, editors, *Information Security and Privacy, Second Australasian Conference, ACISP*, LNCS vol. 1270, pp. 303–313. Springer, 1997.
22. A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. CRYPTO 1984*, LNCS vol. 196, pp. 47–53. Springer, 1984.
23. N.P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, 38(13):630–632, 2002.
24. N.P. Smart. Access control using pairing based cryptography. In M. Joye, editor, *Proceedings CT-RSA 2003*, LNCS vol. 2612, pp. 111–121. Springer, 2003.

Appendix A: Proofs of Security for FullCL-PKE

A.1 Two Public Key Encryption Schemes

We define a public key encryption scheme **HybridPub**. It will be used as a tool in our security proof for FullCL-PKE.

HybridPub: This scheme is specified by three algorithms: **Key-Generation**, **Encrypt** and **Decrypt**.

Key-Generation: (1) Run \mathcal{IG} to generate $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ with the usual properties. Choose a generator $P \in \mathbb{G}_1$. (2) Pick a random $Q \in \mathbb{G}_1^*$, a random $s \in \mathbb{Z}_q^*$ and a random $x \in \mathbb{Z}_q^*$. (3) Set $P_0 = sP$, $X = xP$, $Y = xsP$ and $S = xsQ$. (4) Choose the cryptographic hash functions $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^n$, $H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}_q^*$ and $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$. The public key is $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2, H_3, H_4 \rangle$. The private key is $S = xsQ$, the message space is $\mathcal{M} = \{0, 1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0, 1\}^{2n}$.

Encrypt: To encrypt $M \in \mathcal{M}$, perform the following steps: (1) Check that the equality $e(X, P_0) = e(Y, P)$ holds. If not, output \perp and abort encryption. (2) Choose a random $\sigma \in \{0, 1\}^n$. (3) Set $r = H_3(\sigma, M)$. (4) Compute and output the ciphertext: $C = \langle rP, \sigma \oplus H_2(e(Q, Y)^r), M \oplus H_4(\sigma) \rangle$.

Decrypt: To decrypt $C = \langle U, V, W \rangle \in \mathcal{C}$ using private key S , do the following: (1) Compute $V \oplus H_2(e(S, U)) = \sigma'$. (2) Compute $W \oplus H_4(\sigma') = M'$. (3) Set $r' = H_3(\sigma', M')$ and test if $U = r'P$. If not, output \perp and reject the ciphertext. (4) Output M' as the decryption of C .

A second scheme **BasicPub** is defined in [2]; it is a simplified version of **HybridPub** in which the encryption of message M equals $\langle rP, M \oplus H_2(e(Q, Y)^r) \rangle$. The full paper [2] also defines Type I and II IND-CCA, IND-CPA and OWE adversaries for **BasicPub** and **HybridPub**: these are similar to the usual definitions, except that a Type I adversary is allowed to replace the public key, while a Type II adversary has the value s .

A.2: Statements of Lemmas

We present a series of lemmas. Theorem 1 for Type I adversaries follows by combining Lemmas 2, 3, 4, 5 and 8. Similarly, Theorem 1 for Type II adversaries follows by combining Lemmas 6, 7 and 8.

Lemma 2. *Suppose that H_1, H_2, H_3 and H_4 are random oracles and that there exists an IND-CCA Type I adversary \mathcal{A}_I against FullCL-PKE. Suppose \mathcal{A}_I has advantage ϵ , runs in time t , makes q_i queries to H_i ($1 \leq i \leq 4$) and makes q_d decryption queries. Then there is an algorithm \mathcal{B} which acts as either a Type I or a Type II IND-CPA adversary against HybridPub. Moreover, \mathcal{B} either has advantage at least $\epsilon\lambda^{q_d}/4q_1$ when playing as a Type I adversary, or has advantage at least $\epsilon\lambda^{q_d}/4q_1$ when playing as a Type II adversary. \mathcal{B} runs in time $t + O((q_3 + q_4)q_d t')$. Here t' is the running time of the BasicCL-PKE encryption algorithm and*

$$1 - \lambda \leq (q_3 + q_4) \cdot \epsilon_{\text{OWE}}(t + O((q_3 + q_4)q_d t', q_2) \\ + \epsilon_{\text{GBDHP}}(t + O((q_3 + q_4)q_d t') + 3q^{-1} + 2^{-n+1}),$$

where $\epsilon_{\text{OWE}}(T, q')$ denotes the highest advantage of any Type I or Type II OWE adversary against BasicPub which operates in time T and makes q' hash queries to H_2 , and $\epsilon_{\text{GBDHP}}(T)$ denotes the highest advantage of any time T algorithm to solve GBDHP in groups of order q generated by \mathcal{IG} .

Lemma 3. *Suppose that H_3 and H_4 are random oracles. Let \mathcal{A}_I be a Type I IND-CPA adversary against HybridPub which has advantage ϵ and makes q_4 queries to H_4 . Then there exists a Type I OWE adversary \mathcal{A}'_I against BasicPub which runs in time $O(\text{time}(\mathcal{A}_I))$ and has advantage at least $\epsilon/2(q_3 + q_4)$.*

Lemma 4. *Suppose that H_3 and H_4 are random oracles. Let \mathcal{A}_I be a Type II IND-CPA adversary against HybridPub which has advantage ϵ and makes q_4 queries to H_4 . Then there exists a Type II OWE adversary \mathcal{A}'_I against BasicPub which runs in time $O(\text{time}(\mathcal{A}_{II}))$ and has advantage at least $\epsilon/2(q_3 + q_4)$.*

Lemma 5. *Suppose that H_2 is a random oracle. Suppose there exists a Type I OWE adversary \mathcal{A}_I against BasicPub which makes at most q_2 queries to H_2 and which has advantage ϵ . Then there exists an algorithm \mathcal{B} to solve the GBDHP which runs in time $O(\text{time}(\mathcal{A}_I))$ and has advantage at least $(\epsilon - \frac{1}{2^n})/q_2$.*

Lemma 6. *Suppose that H_1 is a random oracle and that there exists an IND-CCA Type II adversary \mathcal{A}_{II} on FullCL-PKE with advantage ϵ which makes at most q_1 queries to H_1 . Then there is an IND-CCA Type II adversary on HybridPub with advantage at least ϵ/q_1 which runs in time $O(\text{time}(\mathcal{A}_{II}))$.*

Lemma 7. Suppose that H_3 and H_4 are random oracles. Let \mathcal{A}_{II} be a Type II IND-CCA adversary against HybridPub which has advantage ϵ , makes q_d decryption queries, q_3 queries to H_3 and q_4 queries to H_4 . Then there exists a Type II OWE adversary \mathcal{A}'_{II} against BasicPub with

$$\begin{aligned} \text{time}(\mathcal{A}'_{II}) &= \text{time}(\mathcal{A}_{II}) + O(n(q_3 + q_4)) \\ \text{Adv}(\mathcal{A}'_{II}) &\geq \frac{1}{2(q_3 + q_4)} ((\epsilon + 1)(1 - q^{-1} - 2^{-n})^{q_d} - 1). \end{aligned}$$

Lemma 8. Suppose that H_2 is a random oracle. Suppose there exists a Type II OWE adversary \mathcal{A}_{II} against BasicPub which makes at most q_2 queries to H_2 and which has advantage ϵ . Then there exists an algorithm \mathcal{B} to solve the BDHP which runs in time $O(\text{time}(\mathcal{A}_{II}))$ and has advantage at least $(\epsilon - \frac{1}{2^n})/q_2$.

A.3: Proofs of Lemmas

Proof of Lemma 2: Let \mathcal{A}_I be a Type I IND-CCA adversary against FullCL-PKE. Suppose \mathcal{A}_I has advantage ϵ , runs in time t , makes q_i queries to random oracle H_i ($1 \leq i \leq 4$) and makes q_d decryption queries. We show how to construct from \mathcal{A}_I an adversary \mathcal{B} that acts either as a Type I IND-CCA adversary against HybridPub or as a Type II IND-CCA adversary against HybridPub. We assume that challengers $\mathcal{C}_I, \mathcal{C}_{II}$ for both types of game are available to \mathcal{B} .

Adversary \mathcal{B} begins by choosing a random bit c and an index I uniformly at random with $1 \leq I \leq q_1$. If $c = 0$, then \mathcal{B} chooses to play against \mathcal{C}_I and aborts \mathcal{C}_{II} . Here, \mathcal{B} will build a Type I IND-CPA adversary against HybridPub and fails against \mathcal{C}_{II} . When $c = 1$, \mathcal{B} chooses to play against \mathcal{C}_{II} and aborts \mathcal{C}_I . Here, \mathcal{B} will build a Type II IND-CPA adversary against HybridPub and fails against \mathcal{C}_I . In either case, \mathcal{C} will denote the challenger against which \mathcal{B} plays for the remainder of this proof. We let \mathcal{H} denote the event that \mathcal{A}_I chooses ID_I as the challenge identity ID_{ch} . We let \mathcal{F}_0 denote the event that \mathcal{A}_I extracts the partial private key for entity ID_I and \mathcal{F}_1 denote the event that \mathcal{A}_I replaces the public key of entity ID_I at some point in its attack.

If $c = 0$, then \mathcal{C} is a Type I challenger for HybridPub and begins by supplying \mathcal{B} with a public key $K_{\text{pub}} = \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2, H_3, H_4 \rangle$. If $c = 1$, then \mathcal{C} is a Type II challenger and so supplies \mathcal{B} with a public key K_{pub} together with the value s such that $P_0 = sP$. Then \mathcal{B} simulates the algorithm Setup of FullCL-PKE for \mathcal{A}_I by supplying \mathcal{A}_I with $\text{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2, H_3, H_4 \rangle$. Here H_1 is a random oracle that will be controlled by \mathcal{B} . Adversary \mathcal{A}_I may make queries of the random oracles H_i , $1 \leq i \leq 4$, at any time during its attack. These are handled as follows:

H_1 queries: \mathcal{B} maintains a list of tuples $\langle \text{ID}_i, Q_i, b_i, x_i, X_i, Y_i \rangle$ which we call the H_1 list. The list is initially empty, and when \mathcal{A}_I queries H_1 on input $\text{ID} \in \{0, 1\}^*$, \mathcal{B} responds as follows:

(1) If ID already appears on the H_1 list in a tuple $\langle \text{ID}_i, Q_i, b_i, x_i, X_i, Y_i \rangle$, then \mathcal{B} responds with $H_1(\text{ID}) = Q_i \in \mathbb{G}_1^*$.

(2) If ID does not already appear on the list and ID is the I -th distinct H_1 query made by \mathcal{A}_I , then \mathcal{B} picks b_I at random from \mathbb{Z}_q^* , outputs $H(ID) = b_I Q$ and adds the entry $\langle ID, b_I Q, b_I, \perp, X, Y \rangle$ to the H_1 list.

(3) Otherwise, when ID does not already appear on the list and ID is the i -th distinct H_1 query made by \mathcal{A}_I where $i \neq I$, \mathcal{B} picks b_i and x_i at random from \mathbb{Z}_q^* , outputs $H(ID) = b_i P$ and adds $\langle ID, b_i P, b_i, x_i, x_i P, x_i P_0 \rangle$ to the H_1 list.

Notice that with this specification of H_1 , the FullCL-PKE partial private key for ID_i ($i \neq I$) is equal to $b_i P_0$ while the public key for ID_i is $\langle x_i P, x_i P_0 \rangle$ and the private key for ID_i is $x_i b_i P_0$. These can all be computed by \mathcal{B} when $c = 0$. Additionally, when $c = 1$ (so \mathcal{B} has s), \mathcal{B} can compute $sb_I Q$, the partial private key of ID_I .

H_2 queries: Any H_2 queries made by \mathcal{A}_I are passed to \mathcal{C} to answer. We do need to assume in the course of the proof that H_2 is a random oracle.

H_3 and H_4 queries: Adversary \mathcal{B} passes \mathcal{A}_I 's H_3 and H_4 queries to \mathcal{C} to answer, but keeps lists $\langle \sigma_j, M_j, H_{3,j} \rangle$ and $\langle \sigma'_i, H_{4,i} \rangle$ of \mathcal{A}_I 's distinct queries and \mathcal{C} 's replies to them.

Phase 1: After receiving **params** from \mathcal{B} , \mathcal{A}_I launches Phase 1 of its attack, by making a series of requests, each of which is either a partial private key extraction for an entity, a private key extraction for an entity, a request for a public key for an entity, a replacement of a public key for an entity or a decryption query for an entity. We assume that \mathcal{A}_I always makes the appropriate H_1 query on the identity ID for that entity before making one of these requests. \mathcal{B} replies to these requests as follows:

Partial Private Key Extraction: Suppose the request is on ID_i . There are three cases: (1) If $i \neq I$, then \mathcal{B} replies with $b_i P_0$. (2) If $i = I$ and $c = 0$, then \mathcal{B} aborts. (3) If $i = I$ and $c = 1$, then \mathcal{B} replies with $sb_I Q$.

Private Key Extraction: Suppose the request is on ID_i . We can assume that the public key for ID_i has not been replaced. There are two cases: (1) If $i \neq I$, then \mathcal{B} outputs $x_i b_i P_0$. (2) If $i = I$, then \mathcal{B} aborts.

Request for Public Key: If the request is on ID_i then \mathcal{B} returns $\langle X_i, Y_i \rangle$ by accessing the H_1 list.

Replace Public Key: Suppose the request is to replace the public key for ID_i with value $\langle X'_i, Y'_i \rangle$. (We know that this will be a valid public key, i.e. a key satisfying $e(X'_i, P_0) = e(Y'_i, P)$). There are two cases: (1) If $i = I$ and $c = 1$, then \mathcal{B} aborts. (2) Otherwise, \mathcal{B} replaces the current entries X_i, Y_i in the H_1 list with the new entries X'_i, Y'_i . If $i = I$, then \mathcal{B} makes a request to its challenger \mathcal{C} to replace the public key components $\langle X, Y \rangle$ in K_{pub} with new values $\langle X'_I, Y'_I \rangle$.

Decryption Queries: Suppose the request is to decrypt ciphertext $\langle U, V, W \rangle$ for ID_ℓ , where (as discussed in Section 3), the private key that should be used is the one corresponding to the current value of the public key for ID_i . Notice that even when $\ell = I$, \mathcal{B} cannot make use of \mathcal{C} to answer the query, because \mathcal{B} is meant to be an IND-CPA adversary. Instead \mathcal{B} makes use of an algorithm \mathcal{KE} .

Algorithm \mathcal{KE} : The input to the algorithm is a ciphertext $C = \langle U, V, W \rangle$, an identity ID_ℓ and the current value of the public key $\langle X_\ell, Y_\ell \rangle$. We assume that \mathcal{KE} also has access to the H_3 and H_4 lists. \mathcal{KE} operates as follows:

(1) Find all triples $\langle \sigma_j, M_j, H_{3,j} \rangle$ on the H_3 list such that

$$\langle U, V \rangle = \text{BasicCL-PKE-Encrypt}_{ID_\ell, \langle X_\ell, Y_\ell \rangle}(\sigma_j; H_{3,j}).$$

Here, $\text{BasicCL-PKE-Encrypt}_{ID_A, \langle X_A, Y_A \rangle}(M; r)$ denotes the BasicCL-PKE encryption of message M for ID_A using public key $\langle X_A, Y_A \rangle$ and random value r . Collect all these triples in a list S_1 . If S_1 is empty, output \perp and halt.

(2) For each triple $\langle \sigma_j, M_j, H_{3,j} \rangle$ in S_1 , find all pairs $\langle \sigma'_i, H_{4,i} \rangle$ in the H_4 list with $\sigma_j = \sigma'_i$. For each such match, place $\langle \sigma_j, M_j, H_{3,j}, H_{4,i} \rangle$ on a list S_2 . If S_2 is empty, then output \perp and halt.

(3) Check in S_2 for an entry such that $W = M_j \oplus H_{4,i}$. If such an entry exists, then output M_j as the decryption of $\langle U, V, W \rangle$. Otherwise, output \perp .

We prove that \mathcal{KE} correctly decrypts with high probability in Lemma 9.

Challenge Phase: At some point, \mathcal{A}_I should decide to end Phase 1 and pick ID_{ch} and two messages m_0, m_1 on which it wishes to be challenged. We can assume that ID_{ch} has already been queried of H_1 but that \mathcal{A}_I has not extracted the private key for this identity. Algorithm \mathcal{B} responds as follows. If $ID_{ch} \neq ID_I$ then \mathcal{B} aborts. Otherwise $ID_{ch} = ID_I$ and \mathcal{B} gives \mathcal{C} the pair m_0, m_1 as the messages on which it wishes to be challenged. \mathcal{C} responds with the challenge ciphertext $C' = \langle U', V', W' \rangle$, such that C' is the HybridPub encryption of m_b under K_{pub} for a random $b \in \{0, 1\}$. Then \mathcal{B} sets $C^* = \langle b_I^{-1}U', V', W' \rangle$ and delivers C^* to \mathcal{A}_I . It is easy to see that C^* is the FullCL-PKE encryption of m_b for identity ID_I under public key $\langle X_I, Y_I \rangle$. We let $\langle X_{ch}, Y_{ch} \rangle$ denote the particular value of the public key for identity ID_{ch} during the challenge phase (\mathcal{A}_I may change this key in Phase 2 of its attack).

Phase 2: \mathcal{B} continues to respond to \mathcal{A}_I 's requests as in Phase 1.

Guess: Eventually, \mathcal{A}_I should make a guess b' for b . Then \mathcal{B} outputs b' as its guess for b . If \mathcal{A}_I has used more than time t , or attempts to make more than q_i queries to random oracle H_i or more than q_d decryption queries, then \mathcal{B} should abort \mathcal{A}_I and output a random guess for bit b (in this case algorithm \mathcal{KE} has failed to perform correctly at some point).

Analysis: We claim that if algorithm \mathcal{B} does not abort during the simulation and if all of \mathcal{B} 's uses of the algorithm \mathcal{KE} result in correct decryptions, then algorithm \mathcal{A}_I 's view is identical to its view in the real attack. Moreover, if this is the case, then $2(\Pr[b = b'] - \frac{1}{2}) \geq \epsilon$. This is not hard to see: \mathcal{B} 's responses to all hash queries are uniformly and independently distributed as in the real attack. All responses to \mathcal{A}_I 's requests are valid, provided of course that \mathcal{B} does not abort and that \mathcal{KE} performs correctly. Furthermore, the challenge ciphertext C^* is a valid FullCL-PKE encryption of m_b under the current public key for identity ID_{ch} , where $b \in \{0, 1\}$ is random. Thus, by definition of algorithm \mathcal{A}_I we have that $2(\Pr[b = b'] - \frac{1}{2}) \geq \epsilon$.

So we must examine the probability that \mathcal{B} does not abort during the simulation given that the algorithm \mathcal{KE} performs correctly. Examining the simulation, we see that \mathcal{B} can abort for one of four reasons:

- (0) Because $c = 0$ and the event \mathcal{F}_0 occurred during the simulation.
- (1) Because $c = 1$ and event \mathcal{F}_1 occurred during the simulation.
- (2) Because \mathcal{A}_I made a private key extraction on ID_I at some point.
- (3) Or because \mathcal{A}_I chose $\text{ID}_{\text{ch}} \neq \text{ID}_I$.

We name the event $(c = i) \wedge \mathcal{F}_i$ as \mathcal{H}_i for $i = 0, 1$. We also name the last two events here as \mathcal{F}_2 and \mathcal{F}_3 . Of course, \mathcal{F}_3 is the same as event $\neg\mathcal{H}$. Now \mathcal{A}_I makes q_1 queries of H_1 and chooses ID_{ch} from amongst the responses ID_i , while \mathcal{B} 's choice of I is made uniformly at random from the set of q_1 indices i . So the probability that $\text{ID}_{\text{ch}} = \text{ID}_I$ is equal to $1/q_1$. Hence $\Pr[\mathcal{H}] = 1/q_1$. Notice too that the event $\neg\mathcal{F}_3$ implies the event $\neg\mathcal{F}_2$ (if \mathcal{A}_I chooses $\text{ID}_{\text{ch}} = \text{ID}_I$, then no private key extraction on ID_I is allowed). Gathering this information together:

$$\Pr[\mathcal{B} \text{ does not abort}] = \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 \wedge \neg\mathcal{F}_2 \wedge \neg\mathcal{F}_3] = \frac{1}{q_1} \cdot \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 | \mathcal{H}].$$

Notice now that the events \mathcal{H}_0 and \mathcal{H}_1 are mutually exclusive (because one involves $c = 0$ and the other $c = 1$). Therefore we have

$$\Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 | \mathcal{H}] = 1 - \Pr[\mathcal{H}_0 | \mathcal{H}] - \Pr[\mathcal{H}_1 | \mathcal{H}].$$

Moreover, $\Pr[\mathcal{H}_i | \mathcal{H}] = \Pr[(c = i) \wedge \mathcal{F}_i | \mathcal{H}] = \frac{1}{2} \Pr[\mathcal{F}_i | \mathcal{H}]$, where the last equality follows because the event $\mathcal{F}_i | \mathcal{H}$ is independent of the event $c = i$. So we have

$$\Pr[\mathcal{B} \text{ does not abort}] = \frac{1}{q_1} \left(1 - \frac{1}{2} \Pr[\mathcal{F}_0 | \mathcal{H}] - \frac{1}{2} \Pr[\mathcal{F}_1 | \mathcal{H}] \right).$$

Finally, we have that $\Pr[\mathcal{F}_0 \wedge \mathcal{F}_1 | \mathcal{H}] = 0$ because of the rules on adversary behaviour described in Section 3 (an adversary cannot both extract the partial private key and change the public key of the challenge identity). This implies that $\Pr[\mathcal{F}_0 | \mathcal{H}] + \Pr[\mathcal{F}_1 | \mathcal{H}] \leq 1$. Hence we see that $\Pr[\mathcal{B} \text{ does not abort}] \geq 1/2q_1$.

Now we examine the probability that algorithm \mathcal{KE} correctly handles all of \mathcal{A}_I 's q_d decryption queries. We will show in Lemma 9 below that the probability that \mathcal{KE} correctly replies to individual decryption queries is at least λ , where λ is bounded as in the statement of this lemma.

It is now easy to see that \mathcal{B} 's advantage is at least $\frac{\epsilon}{2q_1} \lambda^{q_d}$. It follows that either \mathcal{B} 's advantage as a Type I adversary against **HybridPub** or \mathcal{B} 's advantage as a Type II adversary against **HybridPub** is at least $\frac{\epsilon}{4q_1} \lambda^{q_d}$. The running time of \mathcal{B} is $\text{time}(\mathcal{A}_I) + q_d \cdot \text{time}(\mathcal{KE}) = t + O((q_3 + q_4)q_d t')$ where t' is the running time of the **BasicCL-PKE** encryption algorithm. This completes the proof.

Lemma 9. *In the simulation in the proof of Lemma 2, Algorithm \mathcal{KE} correctly replies to individual decryption queries with probability at least λ where*

$$1 - \lambda \leq (q_3 + q_4) \cdot \epsilon_{\text{OWE}}(t + O((q_3 + q_4)q_d t', q_2) + \epsilon_{\text{GBDHP}}(t + O((q_3 + q_4)q_d t') + 3q^{-1} + 2^{-n+1}).$$

Here t is the running time of adversary \mathcal{A}_I , t' is the running time of the **BasicCL-PKE** encryption algorithm, $\epsilon_{\text{OWE}}(T, q')$ denotes the highest advantage of any Type I or Type II OWE adversary against **BasicPub** which operates in time T and makes q' hash queries to H_2 , and $\epsilon_{\text{GBDHP}}(T)$ denotes the highest advantage

of any algorithm to solve GBDHP in time T in groups of order q generated by \mathcal{IG} .

Proof of Lemma 9: The proof, which is given in [2], is closely modelled on the proof of [11, Lemma 11], but differs in several key respects: we need to build an algorithm which handles multiple public keys, and the algorithm can be asked to decrypt the challenge ciphertext (but under a different identity/public key combination from the challenge identity). This substantially complicates the analysis.

Proof of Lemma 3: This proof is modelled on the proof of [11, Lemma 10], modified to handle \mathcal{A}_I 's ability to replace public keys. See [2] for details.

Proof of Lemma 4: The proof technique is similar to that used in Lemma 3.

Proof of Lemma 5: This proof is similar to that of [5, Theorem 4.1], with modifications to handle adversaries who can replace public keys.

Proof of Lemma 6: The proof is in the full version [2]; it uses ideas from both the $c = 1$ case of the proof of Lemma 2, and the proof of [5, Lemma 4.6].

Proof of Lemma 7: This is easily proven using [11, Theorem 14], noting that s can be made available to Type II adversaries simply by including it in public keys. We also use the fact that HybridPub is $1/q$ -uniform in the sense of [11].

Proof of Lemma 8: The proof in [2] uses similar techniques to the proof of Lemma 5 with a twist to handle the Type II adversary's knowledge of s .

A Complete and Explicit Security Reduction Algorithm for RSA-Based Cryptosystems

Kaoru Kurosawa¹, Katja Schmidt-Samoa², and Tsuyoshi Takagi²

¹ Ibaraki University

4-12-1 Nakanarusawa, Hitachi, Ibaraki, 316-8511, Japan

`kurosawa@cis.ibaraki.ac.jp`

² Technische Universität Darmstadt, Fachbereich Informatik

Alexanderstr.10, D-64283 Darmstadt, Germany

`{samoa,takagi}@informatik.tu-darmstadt.de`

Abstract. In this paper, we introduce a conceptually very simple and demonstrative algorithm for finding small solutions (x, y) of $ax + y = c \bmod N$, where $\gcd(a, N) = 1$. Our new algorithm is a variant of the Euclidian algorithm. Unlike former methods, it finds a small solution whenever such a solution exists. Further it runs in time $\mathcal{O}((\log N)^3)$, which is the same as the best known previous techniques, e.g. lattice-based solutions.

We then apply our algorithm to RSA-OAEP and RSA-Paillier to obtain better security proofs. We believe that there will be many future applications of this algorithm in cryptography.

Keywords: Provable security, Euclidean algorithm, Lattice reduction, RSA cryptosystem.

1 Introduction

Lattice reduction algorithms have been successfully applied to many cases of modern cryptography. Especially, this methods allow us to find a small solution (x, y) of the linear modular congruence

$$ax + y = c \bmod N, \tag{1}$$

where the integers a and N are coprime, i.e. $\gcd(a, N) = 1$. This technique was used to prove the security of RSA-OAEP and RSA-Paillier.

By using the above mentioned technique, Fujisaki et al. showed that RSA-OAEP is semantically secure against adaptive chosen ciphertext attacks (IND-CCA2) under the RSA assumption in the random oracle model [FOPS01] after important works of [BR95, Sho02]. In the random oracle model, the OAEP conversion is a technique to design a secure encryption scheme from any trapdoor one-way permutation [BR95]. We write f -OAEP if f is the underlying trapdoor function. Today's most famous cryptosystem, RSA-OAEP, is a result of this work.

In the standard model, on the other hand, it is known that RSA-Paillier encryption scheme is semantically secure against chosen plaintext attacks (IND-CPA). After the work of [ST02], Catalano et al. proved that the one-wayness of RSA-Paillier is equivalent to that of RSA [CGHGN01] by using the above technique with $c = 0$.

Now it is an important aim in cryptography to improve security reduction proofs, because the proposed size of the security parameters of a cryptosystem is directly influenced by the reduction costs.

In this paper, we introduce a conceptually much simpler and demonstrative algorithm for finding small solutions (x, y) of eq.(1). Our new algorithm is a variant of the Euclidian algorithm. Unlike the lattice-based method, it exploits that the sought-after small solution is non-negative. Further, it runs in time $O((\log N)^3)$, which is the same as the lattice-based method.

We then apply our algorithm to the security proof of RSA-OAEP to enhance the advantage of the reduction algorithm. The proof of RSA-OAEP is divided into two parts [FOPS01]. The first part was to prove the semantic security of the general OAEP conversion scheme under the so-called partial-domain one-wayness of the underlying trapdoor permutation. The second part was to exploit the homomorphic properties of RSA function in order to show the equivalence of partial-domain one-wayness and full-domain one-wayness in the RSA case.

However, the second part does not work for all values of a of eq.(1). More precisely, it works if the lattice $L_{a,N} = \{(u, v) \in \mathbb{Z}^2 \mid au = v \bmod N\}$ contains no non-zero vector of length at most 2^{k_0+2} , where k_0 is the maximal bit-length of the sought-after small solution. Since there are approximately $\pi 2^{2k_0+4} < 2^{2k_0+6}$ lattices containing a non-zero vector shorter than 2^{k_0+2} , the number of bad values for a is bounded above by 2^{2k_0+6} . Obviously, this result is not optimal, especially if the bound k_0 is close to half of the bit-length of N . One reason for the non-optimal performance of the lattice-based method is that it does not exploit all the information given about the sought-after solution. Namely, it takes no advantage of the fact that the solution is non-negative, not only small in absolute value.

For this problem, we are able to upper-bound the number of bad values for a by 2^{2k_0+1} instead of 2^{2k_0+6} .

Finally for RSA-Paillier, we use our new algorithm to construct an alternative reduction proof, extending the important work of Catalano et al. [CNS02]. Based on the analysis of our algorithm, we give the exact security analysis while Catalano et al. gave only asymptotic results.

But we want to point out that the major aim of this paper is not the advancement of the reduction proofs of RSA-OAEP and RSA-Paillier, respectively. Indeed, the achieved improvements are not dramatic ones. In fact, the main objective of this paper is the introduction of a new algorithm for solving two-variable linear congruence with small solutions. We believe that there will be many future applications of this algorithm in cryptography. To confirm this assumption, we revisit the security proofs of RSA-OAEP and RSA-Paillier as two applications.

(Related works:) Note that this task is not a new one in cryptography. In 1985, De Jonge and Chaum developed an attack against some kinds of RSA signature schemes [JC86], which was enlarged in 1997 by Girault and Misarsky [GM97], [Kat01]. These attacks utilize an affine variant of the Euclidian algorithm for solving two-variable linear modular equations with small solutions. But it has to be stressed, that this algorithm may fail, even if small solutions exist.

If $c = 0$, it is possible to find small solutions by means of continued fractions. Again, the Euclidian algorithm is used. But as before, this method is only heuristic, i.e. it does not succeed with all input.

Our algorithm, on the contrary, works for arbitrary inputs.

This paper is organized as follows: In Section 2 the security reduction algorithms of the RSA-OAEP and the RSA-Paillier cryptosystem are reviewed. In Section 3 we present our proposed algorithm for solving a two-variable modular equation with small solutions. In Section 4 the proposed algorithm is applied to the RSA-OAEP and the RSA-Paillier cryptosystem. In Section 5 we state a concluding remark.

2 Security Reduction Algorithms of RSA-OAEP and RSA-Paillier

In this section, we review the reduction proofs of the semantic security of RSA-OAEP and the one-wayness of RSA-Paillier. In both cases we are confronted with the problem of finding small solutions of modular congruences. We sketch the existing solutions which utilize lattice reduction methods.

2.1 RSA-OAEP

Let $f : \{0, 1\}^k \mapsto \{0, 1\}^k$ be a one-way trapdoor permutation. The random oracle reduction proof of f -OAEP states that if there is a CCA2-adversary against the semantic security of f -OAEP with a non-negligible advantage and running time t , then we are able to construct an algorithm \mathcal{A} with the following abilities: On the input $f(s_1, s_2)$, \mathcal{A} computes in time polynomial in t and in the number of the adversary's queries to the different oracles (decryption and hash) a set S , such that the probability of s_1 being an element of S is non-negligible, too. In few words, the semantic security of f -OAEP in the random oracle model is reduced to the partial-domain one-wayness of f .

Now we consider the case $f = \text{RSA}$. We will sketch how the partial-domain one-wayness of RSA is reduced to its full-domain pendant. First, we introduce some notations. If x is a natural number, we write $[x]_l^l$ for the l most significant bits and $[x]_l$ for the l least significant bits of the binary representation of x , respectively. Let N be a k -bit RSA modulus and $k_0 < k/2$. Suppose there is an algorithm \mathcal{A} that on the input $C = m^e \bmod N$ returns a set S of size q containing the integer $x := [m]^{k-k_0}$. We show how to solve the RSA problem (compute m from $C = m^e \bmod N$) using \mathcal{A} as a subroutine. Pick any $a \in \mathbb{Z}_N^\times$ at random and run \mathcal{A} on the inputs C and $C' := Ca^e \bmod N$. Because of the

homomorphic properties of the RSA function we know that C' is the encryption of $ma \bmod N$. Hence the two output-sets produced by \mathcal{A} contain the $k - k_0$ most significant bits of m and $ma \bmod N$, respectively. We define $u := [m]^{k-k_0}_{k_0}$, $r := [m]_{k_0}$, $v := [ma \bmod N]^{k-k_0}_{k_0}$ and $s := [ma \bmod N]_{k_0}$. Thus, $m = u \cdot 2^{k_0} + r$ and $ma \bmod N = v \cdot 2^{k_0} + s$ holds, leading to

$$\begin{aligned} v \cdot 2^{k_0} + s &= a \cdot (u \cdot 2^{k_0} + r) \bmod N \\ \Rightarrow ar &= s + c \bmod N, \quad c = (v - ua) \cdot 2^{k_0} \bmod N. \end{aligned} \quad (2)$$

Thus for each of the q^2 possible combinations u, v taken from the output-sets of the two \mathcal{A} -runs, we get a linear modular congruence in the two unknowns r and s , where $0 \leq r, s < 2^{k_0} < \sqrt{N}$. Note that therefore the reduction cost is quadratic in q (the value q arises in the random oracle part of the RSA-OAEP security proof, namely q equals the number of \mathcal{A}_{SS} 's queries to one of the hash oracles, where \mathcal{A}_{SS} is an adversary against the semantic security of the OAEP conversion scheme). This is the main reason why the RSA-OAEP security proof is not meaningful for real-life parameters. Of course, an improvement of the congruence-solving-step will not affect this problem. Hence it is an important future task to find a reduction proof where only one \mathcal{A} -run is needed.

In the following, we call x, y a *small* solution of the congruence (2) iff $0 \leq x, y < 2^{k_0}$ holds. We explain how Fujisaki et al. find a small solution using the Gaussian reduction algorithm. This algorithm can be viewed as a generalization of the Euclidian algorithm in dimension 2. For all results concerning lattice theory see [MG02], [SF]. At first, compute a reduced basis (U, V) of the lattice $L_{a,N} = \{(x, y) \in \mathbb{Z}^2 \mid ax = y \bmod N\}$ using the Gaussian algorithm. As we can easily find a sufficiently short basis of $L_{a,N}$, for example take the vectors $(1, a)$ and $(1, a + N)$, this can be done in time $\mathcal{O}((\log N)^3)$. Let T be a small solution and T_0 be any solution of (2). To find $T_0 = (x_0, y_0)$, we can choose x_0 as we like and then compute $y_0 = ax_0 - c \bmod N$. Define $l = 2^{k_0+2}$ and assume that $L_{a,N}$ is a so called *l-good* lattice, meaning that there exists no non-zero lattice vector shorter than l . This choice of l together with the properties of a reduced basis guarantee two important facts: in the first place, T is unique as a small solution of (2). Secondly, the coefficients of T in the basis (U, V) are smaller than $1/2$ in absolute value. Thus, the coefficients (in (U, V)) of the lattice point $T - T_0$ can be constructed simply by taking the closest integers to the coefficients of $-T_0$. This is a consequence of the uniqueness of basis representation. From knowledge of T_0 and $T - T_0$, we can easily construct T .

But as stated above, this method only works if the randomly chosen a yields an *l-good* lattice. We already have seen that the absolute number of bad values for a can be bounded above by 2^{2k_0+6} , consequently the probability of choosing a bad value is smaller than 2^{2k_0+6-k} . The total advantage of this reduction is therefore greater than $\varepsilon' = \varepsilon(\varepsilon - 2^{2k_0+6-k})$, where ε denotes the advantage of the partial inverter \mathcal{A} . Note that ε' is non-negligible in $k = \log N$, if ε is non-negligible in k and if k_0 is adequate smaller than k , i.e. there is a rational number $0 < t < 1/2$ such that $k_0 < tk$.

2.2 RSA-Paillier

Let N, e be the RSA public-key. The Hensel lifting problem of the RSA encryption function is to compute $r^e \bmod N^2$ for a given ciphertext $r^e \bmod N$. In 2002, Sakurai and Takagi proved that RSA-Paillier is one-way iff the Hensel-lifting problem is hard [ST02]. Moreover, they introduced a reduction algorithm for solving the RSA problem using the Hensel-lifting oracle as a subroutine. But this algorithm was not efficient (i.e. for each bit of the secret message two oracle-calls were needed), and it could be proven to achieve a non-negligible advantage only in case of a perfect Hensel-lifting oracle. A short time later, Catalano et al. were able to show that the RSA problem could be solved by calling the (potentially non-perfect) Hensel-lifting oracle only twice [CNS02], hence they reduced the one-wayness of RSA-Paillier to the RSA problem. We shortly explain their technique in the following.

Assume that a random RSA ciphertext $c = r^e \bmod N$ is given. We construct an algorithm that computes r given c, N, e using the Hensel lifting. The algorithm obtains $r^e \bmod N^2$ by invoking the Hensel lifting oracle. Then it computes $a^e r^e \bmod N$ for randomly chosen integer $a \in (\mathbb{Z}/N\mathbb{Z})^\times$, and obtains $\mu^e \bmod N^2$ from the Hensel lifting oracle, where $\mu = ar \bmod N$. There is an integer z such that $ar = \mu(1 + zN) \bmod N^2$. The integer $z \bmod N$ can be computed due to $a^e r^e = \mu^e(1 + ezN) \bmod N^2$. Consider the two-dimensional lattice $L = \{(R, U) \in \mathbb{Z}^2 \mid aR = U(1 + zN) \bmod N^2\}$. By the lattice reduction algorithm we can find a vector $(r', \mu') \in L \cap [1, \dots, N-1]^2$ in polynomial time of $\log N$. As the sought-after vector (r, μ) is an element of L , too, we have the relationship $r'\mu = r\mu' \bmod N^2$. Moreover, due to the size constraints $0 < r, r', \mu, \mu' < N$ we conclude that in fact equality holds, i.e. $r'\mu = r\mu'$.

Thus, r and μ are multiples of $r'/\gcd(r', \mu')$ and $\mu'/\gcd(r', \mu')$, respectively, with a factor that is given by $\gcd(r, \mu)$. As with overwhelming probability this factor is sufficiently small, it can be found efficiently by an exhaustive search.

Catalano et al. showed that their method works in time polynomial in $\log N$ with a non-negligible advantage, but they gave no concrete bounds.

3 The Proposed Reduction Algorithm

Let N be a natural number, $0 < a < N$, $0 \leq c < N$, and $\gcd(a, N) = 1$. In this section we give the outline of the algorithm `Lin_Cong` for finding small solutions of the two-variable linear modular congruence

$$ax = y + c \bmod N. \quad (3)$$

To be more concrete, we introduce an algorithm for finding so-called *x-minimal* solutions of (3).

Definition 1. *The pair $(\hat{x}, \hat{y}), 0 \leq \hat{x} < N, 0 \leq \hat{y} < B$ is called a *x-minimal* solution of (3) with respect to the bound $B, 0 < B < N$, if (\hat{x}, \hat{y}) possesses the following properties:*

1. $a\hat{x} = \hat{y} + c \bmod N$.
2. \hat{x} fulfills the following minimality condition: If (x_{alt}, y_{alt}) is a solution of the congruence (3) where $0 \leq y_{alt} < B$ holds, then we have $\hat{x} \leq x_{alt}$.

Note that due to the condition $\gcd(a, N) = 1$ for each B there is exactly one x -minimal solution of (3) w.r.t. B .

As a second step, we propose an efficient variant of the algorithm with complexity $\mathcal{O}((\log N)^3)$. One application of the new algorithm is to replace the lattice based methods used in the reduction proofs described above. Note that we always use $\{0, 1, \dots, N-1\}$ as representatives for the residue classes modulo N . The outline of the proposed algorithm is as follows:

Lin_Cong (Outline)

Input: a, c, N, B , where $0 < a, B < N$, $0 \leq c < N$, and $\gcd(a, N) = 1$

Output: \hat{x}, \hat{y} such that $a\hat{x} = \hat{y} + c \bmod N$ and $\hat{x} \geq 0$ is minimal
with respect to the property that $0 \leq \hat{y} < B$

1. set $a' = a, c' = c, N' = N$
 2. set $y' = -c' \bmod N'$
 3. while $y' \geq B$ do
 4. set $(a', N') = (-N' \bmod a', a')$ (parallel assignment)
 5. set $c' = c' \bmod N', y' = -c' \bmod N'$
 6. set $\hat{y} = y', \hat{x} = a^{-1} \cdot (\hat{y} + c) \bmod N$
 7. return (\hat{x}, \hat{y})
-

In the following, we describe the idea of the proposed algorithm.

First note that $\gcd(a', N') = \gcd(a, N) = 1$ and $a' < N'$ holds in any iteration. Therefore we see that $a' = 0$ is only possible if the corresponding N' (the old value a') equals 1. If this is the case, in step 5 of this iteration we compute $y' = 0$ and the algorithm will terminate. Consequently, the assertion $a' = -N' \bmod a'$ is always defined.

Let (\hat{x}, \hat{y}) be the unique x -minimal solution of (3) w.r.t. B . We show that the algorithm **Lin_Cong (Outline)** on the inputs a, c, N, B returns (\hat{x}, \hat{y}) . To be more precise, the algorithm finds \hat{y} and then computes the corresponding $\hat{x} = a^{-1} \cdot (\hat{y} + c) \bmod N$. The main idea of the algorithm is to reduce the original problem to a smaller instance and iterating this process. This is done as follows: From $a\hat{x} = \hat{y} + c \bmod N$ we deduce $a\hat{x} = \hat{y} + c + kN$ for a suitable $k \in \mathbb{Z}$. Euclidian division yields $N = aq + r$ with $0 \leq r < a$ and a positive integer q . Hence we have

$$\begin{aligned} a\hat{x} = \hat{y} + c + kN &= \hat{y} + c + k(aq + r) \Rightarrow -rk = \hat{y} + c + a(kq - \hat{x}) \\ &\Rightarrow -rk = \hat{y} + c \bmod a \end{aligned}$$

Therefore we have constructed a new linear modular congruence with the new module a in the role of N and the new factor $-r = -N \bmod a$ in the role of a . A solution of this new congruence is given by $(k, \hat{y}) = (\frac{a\hat{x} - \hat{y} - c}{N}, \hat{y})$. The crucial point is the fact that this solution is the x -minimal solution w.r.t. B of the new congruence.

We define the following sequences by iterating this process:

$$\begin{array}{llll} N_0 = N & a_0 = a & c_0 = c & x_0 = \hat{x} \\ N_{i+1} = a_i & a_{i+1} = -N_i \bmod a_i & c_{i+1} = c_i \bmod N_{i+1} & x_{i+1} = \frac{a_i x_i - \hat{y} - c_i}{N_i} \end{array}$$

Note that the first three columns exactly describe the corresponding sequences produced by the algorithm `Lin_Cong (Outline)`. For this reason, we denote by $f_{\text{Lin_Cong}}$ the transformation $(N_i, a_i, c_i) \mapsto (N_{i+1}, a_{i+1}, c_{i+1})$. Let us write cong_i for the linear modular congruence defined with the parameters a_i, c_i and N_i . Inductively, we conclude that the value x_i occurring in the last column leads to a solution (x_i, \hat{y}) of cong_i . Moreover, we can deduce the following lemma (for the rather technical proof see Appendix A):

Lemma 1. *Let (x_i, \hat{y}) be the x -minimal solution of cong_i w.r.t. B and let $x_i > 0$. Then (x_{i+1}, \hat{y}) is the x -minimal solution of cong_{i+1} w.r.t. B . In particular, the y -value of the current x -minimal solution w.r.t. B does not change during the transformation $f_{\text{Lin_Cong}}$, as long as x_i is non-negative.*

Hence with each iteration of the while loop the transformation $f_{\text{Lin_Cong}}$ constructs a smaller problem, because the sequence of the moduli N_i is strictly monotone decreasing. The problem of finding the x -minimal solution is trivial in the following case:

Definition 2. *Let a, c, N, B be integers, where $0 < a, B < N$, $0 \leq c < N$, and $\gcd(a, N) = 1$ hold. The congruence $ax = y + c \bmod N$ satisfies the zero-minimum condition with respect to B , if $-c \bmod N < B$ holds.*

In fact, it is an easy observation that the x -minimal solution of the congruence $ax = y + c \bmod N$ w.r.t. B is given by the pair $(0, -c \bmod N)$ iff $ax = y + c \bmod N$ satisfies the zero-minimum condition w.r.t. B . The aim of the algorithm `Lin_Cong (Outline)` is to convert the original congruence into a congruence satisfying the zero-minimum condition w.r.t. B . This is done using the transformation $f_{\text{Lin_Cong}}$, which does not affect the y -value of the current x -minimal solution w.r.t. B .

Indeed, we can prove the correctness of algorithm `Lin_Cong (Outline)`:

Theorem 1. *Algorithm `Lin_Cong (Outline)` is correct, i.e. given integers a, c, N, B , where $0 < a, B < N$, $0 \leq c < N$, and $\gcd(a, N) = 1$ holds, the algorithm terminates and outputs the unique x -minimal solution \hat{x}, \hat{y} of the congruence $ax = y + c \bmod N$ with respect to the bound B (see Definition 1).*

Proof. Let y_i denote the y -value computed by the algorithm `Lin_Cong (Outline)` in the i th iteration of the while loop. Note that per definition this value yields the solution $(0, y_i)$ of cong_i . For each $i = 0, 1, 2, \dots$ the following holds: Either cong_i satisfies the zero-minimum condition w.r.t. B and consequently $(0, y_i)$ is the x -minimal solution of cong_i w.r.t. B . Or x_i , the x -value of the x -minimal solution of cong_i , is greater zero and lemma 1 tells us that (x_{i+1}, \hat{y}) equals the x -minimal solution of cong_{i+1} w.r.t. B . As the sequence of the moduli N_i is strictly monotone decreasing, there must be an $i \geq 0$ such that cong_i satisfies

the zero-minimum condition w.r.t. B . If this iteration is reached (i.e. we have $y_i < B$ for the first time), then $(0, y_i) = (x_i, \hat{y})$ must hold because according to lemma 1 we know that the y -value of the x -minimal solution w.r.t. B has not changed. Obviously, the x -value computed in step 6 is the correct one.

Analyzing algorithm **Lin_Cong (Outline)** we see that the parallel assignment in step 4 describes a variant of the Euclidian algorithm (set $(a, b) = (-b \bmod a, a)$ instead of set $(a, b) = (b \bmod a, a)$ for $a \leq b$). Obviously, the result remains the same, but unfortunately the variant is less efficient. In particular, in the worst case we need $a - 1$ steps (to see this, try $a = b - 1$), which is by far not fast enough. But some modifications may be helpful: A closer look at the recursion formula $(a, b) = (-b \bmod a, a)$ discloses, that problems occur if $b - a \ll a$ holds. In the following steps the difference $b - a$ is subtracted from a and b until the resulting a is smaller than $b - a$. This procedure may take a long (too long) time. Its result will be $(a \bmod (b - a), b - k(b - a))$, where k equals $a \div (b - a)$ ¹. Therefore we gain a notable speedup by the following case differentiation:

if $b - a \geq a$ then set $(a, b) = (-b \bmod a, a)$
 else set $(a, b) = (a \bmod b - a, b - k(b - a))$ with $k = a \div (b - a)$.

But we need to be a little careful if we wish to assign this idea to the original algorithm (with a' in the role of a and N' in the role of b). In detail, we must not ignore a reduction of the value c' which would have occurred in one of the skipped steps. A possible way out is to skip fewer steps, i.e. we subtract $N' - a'$ until the resulting a' is smaller than $N' - a'$ or c' is greater than the resulting N' . We will see in a while that these modifications are good enough to yield a polynomial running time (in $\log N$). But before doing so, we have to face a last problem: It is possible that the value \hat{y} we are seeking for would be computed in one of the skipped steps. Note that in each skipped step the value $y' = -c' \bmod N'$ is reduced by the amount $N' - a'$ (this is true because due to the above considerations the value c' remains constant). Hence if the resulting y' exceeds the bound B , all the “invisible” values y' computed during the skipped steps do so, too. This means that it is possible to miss the sought-after value \hat{y} only in the last while cycle before termination. So we avoid missing the correct \hat{y} by doing the following: If steps have been skipped during the last while cycle add $N' - a'$ to the current value y' until $y' + k(N' - a')$ exceeds B for the first time. Then set $\hat{y} = y' + (k - 1)(N' - a')$ and compute the corresponding \hat{x} -value as usual.

3.1 Algorithm **Lin_Cong**

The proposed algorithm **Lin_Cong** is as follows:

¹ $x \div y$ denotes the Euclidian quotient of x and y

Lin_Cong

Input: a, c, N, B , where $0 < a, B < N$, $0 \leq c < N$, and $\gcd(a, N) = 1$

Output: \hat{x}, \hat{y} such that $a\hat{x} = \hat{y} + c \pmod N$ and $\hat{x} \geq 0$ is minimal
with respect to the property that $0 \leq \hat{y} < B$

1. set $a' = a, c' = c, N' = N$
 2. set $y' = -c' \pmod{N'}$
 3. while $y' \geq B$ do
 4. set $\text{diff} = N' - a'$
 5. if $\text{diff} < a'$ and $\text{diff} < N' - c'$
 6. then set $k = \min(a' \div \text{diff}, (N' - c') \div \text{diff})$
 7. set $(a', N') = (a' - k \cdot \text{diff}, N' - k \cdot \text{diff})$, set $\text{flag} = 1$
 8. else set $(a', N') = (-N' \pmod{a'}, a')$, set $\text{flag} = 0$
 9. set $c' = c' \pmod{N'}$, set $y' = -c' \pmod{N'}$
 10. If $\text{flag} = 1$ then set $k = \left\lceil \frac{B - y'}{\text{diff}} \right\rceil - 1$, set $\hat{y} = y' + k \cdot \text{diff}$
 11. else set $\hat{y} = y'$
 12. set $\hat{x} = a^{-1} \cdot (\hat{y} + c) \pmod N$
 13. return (\hat{x}, \hat{y})
-

We can prove the following theorem:

Theorem 2. a) The complexity of the algorithm *Lin_Cong* is $\mathcal{O}((\log N)^3)$.

b) Let a, c, N, B be integers, where $0 < a, B < N$, $0 \leq c < N$, and $\gcd(a, n) = 1$ holds. Algorithm *Lin_Cong* on the inputs a, c, N, B finds a small solution $0 \leq \hat{x}, \hat{y} < B$ of the congruence $ax = y + c \pmod N$, provided such a solution exists at all.

Proof. From the discussion above we know that on each input the algorithm *Lin_Cong* computes the same output as its slower variant *Lin_Cong (Outline)*. Thus the second part of the theorem is an immediate consequence of theorem 1. So it remains to show that algorithm *Lin_Cong* runs in polynomial time. We distinguish four cases

1. The condition in step 5 is not fulfilled due to $N' - a' = \text{diff} \geq a'$. Hence the else-case in step 8 is entered. From $N' \geq 2a'$ we deduce that the assignment $N' = a'$ at least halves the value of N' .
2. The condition in step 5 is not fulfilled due to $N' - a' = \text{diff} \geq N' - c'$. Hence the else-case in step 8 is entered and N' is assigned to a' . Because of $a' \leq c'$ the reduction of c' modulo $N' (= a')$ in step 9 at least halves the value of c' .
3. The condition in step 5 is fulfilled and the value k computed in step 6 equals $a' \div (N' - a')$. In this case, the assignment $a' = a' - k \cdot (N' - a')$ done in step 7 is equivalent to $a' = a' \pmod{(N' - a')}$. As we have $N' - a' < a'$, this assignment at least halves the value of a' .
4. The condition in step 5 is fulfilled and the value k computed in step 6 equals $(N' - c) \div (N' - a')$. The value of k is chosen in order to achieve that $N' - (k + 1)(N' - a') \leq c'$ holds. Hence the reduction of c' modulo N' in step 9 of the following while cycle at least halves the value of c' .

Summing up, we see that at least in each second while cycle at least one of the values a', c' and N' is at least halved. Note that the algorithm terminates at once if $a' = 0, c' = 0$ or $N' = 1$ holds. So the number of while cycles is bounded above by $\log a + 2 \log c + \log N$. Each step during the while loop can be done in $\mathcal{O}((\log N)^2)$, therefore the time complexity of the algorithm `Lin_Cong` is $\mathcal{O}((\log N)^3)$.

3.2 Finding All Small Solutions

In this subsection we show that algorithm `Lin_Cong` can be modified to find all small solutions (x, y) of the linear modular congruence (3). We call (x, y) a *small* solution, if $0 \leq x, y < \sqrt{N}$ is satisfied. The time needed for computing all small solutions is $\mathcal{O}((\log N)^3) + l\mathcal{O}(\log N)$, where l is the absolute number of small solutions. The most important observation is that there is a quite simple relationship between all the small solutions in the case of $c = 0$. The general case $c \neq 0$ can be easily derived from the special case. We will see that in both cases all small solutions are located on the same line.

The Case $c = 0$. Let (x_0, y_0) and (x_1, y_1) be two different small solutions of

$$ax = y \bmod N, \gcd(a, N) = 1, \quad (4)$$

i.e. we have

$$ax_0 = y_0 \bmod N \text{ and } ax_1 = y_1 \bmod N,$$

leading to

$$x_0y_1 = x_1y_0 \bmod N.$$

But due to the size-constraints we deduce that this relationship even holds in \mathbb{Z} . Consequently, all small solutions are located on the same line through the origin. Hence to get all small solutions we simply have to compute all integer multiples $(k\hat{x}, k\hat{y}), k \in \mathbb{Z}^{\geq 0}, k\hat{x}, k\hat{y} < \sqrt{N}$, where (\hat{x}, \hat{y}) is the smallest non-zero solution of (4). This solution can be obtained using algorithm `Lin_Cong`. If we run algorithm `Lin_Cong` on an input with $c = 0$, then it will terminate at once with the result $(0, 0)$. But we are seeking for a non-zero solution, hence we exploit the relationship $ax = y \bmod N \Leftrightarrow a(x - 1) = y - a \bmod N$. Namely, we run `Lin_Cong` on the input $(a, N - a, N, \sqrt{N})$, get the result (x', y') , and return $(\hat{x}, \hat{y}) := (x' + 1, y')$. Theorem 1 shows that (\hat{x}, \hat{y}) indeed yields the smallest non-zero solution of (4).

The Case $c \neq 0$. Let (\hat{x}, \hat{y}) be the small solution computed by algorithm `Lin_Cong` on the input (a, c, N, \sqrt{N}) and let (x_{alt}, y_{alt}) be a different small solution. In particular, the difference $(x_{alt} - \hat{x}, y_{alt} - \hat{y})$ is a non-zero solution of (4). As \hat{x} is minimal, we know $\hat{x} < x_{alt}$. Thus we conclude $0 < x_{alt} - \hat{x} < \sqrt{N}, -\sqrt{N} < y_{alt} - \hat{y} < \sqrt{N}$. We distinguish two cases:

1. If $y_{alt} > \hat{y}$ holds, then $(x_{alt} - \hat{x}, y_{alt} - \hat{y})$ is a small solution of (4) and can be found as described above.
2. Otherwise $(x_{alt} - \hat{x}, y_{alt} - \hat{y})$ is a solution of (4), too, but only small in absolute value (with a negative y -component). It is easy to see that we can find all solutions (x, y) , $0 \leq x < \sqrt{N}$, $-\sqrt{N} < y \leq 0$ of (4) by computing all small solutions of $(-a)x = y \bmod N$ as usual and then changing the signs of the y -components.

Note that at most one of these two cases may appear, because if there are two additional small solutions (x_{alt1}, y_{alt1}) and (x_{alt2}, y_{alt2}) with $y_{alt1} > \hat{y}$ and $y_{alt2} < \hat{y}$, then the three differences $(x_{alt1} - \hat{x}, y_{alt1} - \hat{y})$, $(x_{alt2} - \hat{x}, y_{alt2} - \hat{y})$ and $(x_{alt1} - x_{alt2}, y_{alt1} - y_{alt2})$ must be located on at most two lines through the origin, a contradiction.

This leads to the following algorithm:

Lin-Cong-All

Input: a, c, N , where $0 < a, c < N$, and $\gcd(a, N) = 1$

Output: Set $S = \{(x, y) | ax = y + c \bmod N, 0 \leq x, y < \sqrt{N}\}$

1. set $S = \{\}$.
 2. set $(\hat{x}, \hat{y}) = \text{Lin-Cong}(a, c, N, \sqrt{N})$
 3. if $\hat{x} \geq \sqrt{N}$ then return S and stop
 4. else append (\hat{x}, \hat{y}) to S
 5. set $(x', y') = \text{Lin-Cong}(a, N - a, N, \sqrt{N})$
 6. set $(x_0, y_0) = (x' + 1, y')$, set $k = 1$
 7. while $\hat{x} + kx_0 < \sqrt{N}$ and $\hat{y} + ky_0 < \sqrt{N}$ do
 8. append $(\hat{x} + kx_0, \hat{y} + ky_0)$ to S and increment k
 9. if $\#S > 1$ then return S and stop
 10. set $(x', y') = \text{Lin-Cong}(N - a, a, N, \sqrt{N})$
 11. set $(x_0, y_0) = (x' + 1, -y')$, set $k = 1$
 12. while $\hat{x} + kx_0 < \sqrt{N}$ and $\hat{y} + ky_0 \geq 0$ do
 13. append $(\hat{x} + kx_0, \hat{y} + ky_0)$ to S and increment k
 14. return S
-

In step 2, we use algorithm **Lin-Cong** to compute the small solution with the minimal x -coordinate \hat{x} . If even \hat{x} exceeds the bound \sqrt{N} , then obviously no small solution exists at all. The value (x_0, y_0) computed in step 6 equals the smallest non-zero solution of (4). As we have seen above, each sum of (\hat{x}, \hat{y}) and an integer multiple of (x_0, y_0) yields a solution of $ax = y + c \bmod N$. But as \hat{x} is minimal, we only have to consider factors $k \geq 1$. If there is at least one small solution $(\hat{x} + kx_0, \hat{y} + ky_0)$, we know that all small solutions have to be of this shape. Hence the while loop in step 7 and 8 has finds all remaining small solutions and the algorithm terminates. Otherwise we compute the smallest non-zero solution of $ax = y \bmod N$ with a negative y -component (step 10 and 11) and proceed in the same way as before.

3.3 Comparison with the Continuous Fraction Method

Another often used method for finding small solutions of linear modular congruence where the affine coefficient c equals zero is obtained by the continued fraction expansion. We call this method the Euclidean reduction (See [HW79] for the comprehensive treatment). To resume, this method finds all fractions $\frac{p}{q}$ nearby a rational number α (i.e. we have $|\alpha - \frac{p}{q}| < \frac{1}{2q^2}$), where the fractions $\frac{p}{q}$ come in their lowest terms. Assume that we want to find small solutions that do not exceed \sqrt{N} of the congruence $ax = y \bmod N$, where $\gcd(a, N) = 1$ holds. As we have already shown in subsection 3.2, all these solutions are located on the same line through the origin. Therefore there exists a solution (\hat{x}, \hat{y}) such that $\gcd(\hat{x}, \hat{y}) = 1$ is fulfilled. From $a\hat{x} = \hat{y} \bmod N$ we conclude that there is an integer k such that $a\hat{x} = \hat{y} + kN$. We have

$$\frac{a}{N} - \frac{k}{\hat{x}} = \frac{\hat{y}}{N\hat{x}}. \quad (5)$$

If $2\hat{x}\hat{y} < N$ holds, the upper-bound of $\hat{y}/N\hat{x}$ is $1/2\hat{x}^2$. If in addition the rational number k/\hat{x} is irreducible, i.e. $\gcd(k, \hat{x}) = 1$, we can find the integer \hat{x} and thus \hat{y} by using the Euclidian reduction method. Note that $\gcd(k, \hat{x}) = 1$ holds because $\gcd(\hat{x}, \hat{y}) = 1$ is satisfied. If $\gcd(k, \hat{x}) = 1$ is not true, there is an integer $\delta > 1$ such that $\gcd(k, \hat{x}) = \delta$. From $a\hat{x} - Nk = \hat{y}$, we have $\delta|\hat{y}$ and hence $\delta|\gcd(\hat{x}, \hat{y})$. It contradicts to $\gcd(\hat{x}, \hat{y}) = 1$. Summing up, we can use this method if we know that the product $2\hat{x}\hat{y}$ does not exceed N . Consequently, we prefer the use of algorithm `Lin_Cong`, which finds \hat{x}, \hat{y} , even if $2\hat{x}\hat{y} < N$ is not fulfilled.

4 Security Reduction Analysis Using the Proposed Algorithm

In this section, we show how algorithm `Lin_Cong` may be applied to the reduction proofs of RSA-OAEP and RSA-Paillier. In the case of RSA-OAEP we will upper-bound the number of bad values a by 2^{2k_0+1} , compared to the former bound 2^{2k_0+6} . Regarding to RSA-Paillier, we will give an explicit reduction algorithm based on the work of Catalano et al. [CNS02]. We will achieve reduction time $2t + \mathcal{O}((\log N)^3 \varepsilon^{-2})$ and advantage $\varepsilon' > \varepsilon^2/5$ where t and ε are the time and the advantage of the Hensel-lifting oracle, respectively.

4.1 Application to RSA-OAEP

In section 2.1 we have described the reduction proof given by Fujisaki et al. [FOPS01]. Remember that they have constructed the following congruence

$$ax = y + c \bmod N, \quad c = (v - ua) \cdot 2^{k_0} \bmod N, \quad (6)$$

where u and v are built of the $k - k_0$ most significant bits of m or $ma \bmod N$, respectively. In the RSA-OAEP case we call (x, y) a *small* solution of the

congruence (6) iff $0 \leq x, y < 2^{k_0}$ holds. The congruence (6) is known to have the small solution (r, s) , where r is built from the remaining k_0 least significant bits of m .

In section 2.1 we have already seen that the lattice based method only works if the randomly chosen value a yields an l -good lattice. In contrast, algorithm `Lin_Cong` always finds a small solution, provided a small solution exists at all. But it has to be stressed, that referring to the lattice method the choice of a good value a ensures that there exists exactly one small solution. This is an important property, because if the small solution (r, s) is not unique, there is of course no warrant that the solution computed with our algorithm is the correct one. A possible way out is to use algorithm `Lin_Cong_All` instead, which computes all small solutions, and to test each of them. But this is only efficient if the set of small solutions is not too big. Let l be a “sufficiently” small natural number. We want to bound above the probability that the number of small solutions does not exceed l . As we have seen in subsection 3.2, each small solution is of the shape $(\hat{x} + kx_0, \hat{y} + ky_0)$, where (\hat{x}, \hat{y}) is the special solution computed by the algorithm `Lin_Cong` and (x_0, y_0) is either the shortest element of $\{(x, y) | ax = y \bmod N, 0 < x, y < 2^{k_0}\}$ or (x_0, y_0) is the shortest element of $\{(x, y) | ax = y \bmod N, 0 < x < 2^{k_0}, -2^{k_0} < y < 0\}$. Hence there are at most l small solutions of (6), iff the congruence $ax = y \bmod N$ has no solution (x, y) , where

$$0 < x < 2^{k_0}/l, -2^{k_0}/l < y < 2^{k_0}/l. \quad (7)$$

We call a a *bad value*, if there exists a solution of $ax = y \bmod N$ fulfilling the size constraints (7). If (7) holds for (x, y) , then there is exactly one a such that (x, y) is a solution of $ax = y \bmod N$, namely $a = x^{-1}y \bmod N$. Note that due to the size constraints no problems of computing modular inverses occur. Hence there are at most $2^{2k_0+1}/l^2$ bad values of a . The maximal number is 2^{2k_0+1} for $l = 1$.

Therefore, in case of using the lattice solution the probability to choose a bad value a is at least 2^5 times greater compared with the corresponding probability in case of using algorithm `Lin_Cong_All`. We finish with the following theorem:

Theorem 3. *Assume there is an adversary that on input $N, e, m^e \bmod N$ returns the $k - k_0$ most significant bits of m with advantage ε and in time t , where N is a k -bit RSA modulus, e is a public key belonging to N and $2k_0 < k$ holds. Let $l \leq (\log N)^2$ be any natural number. Then with advantage $\varepsilon' > \varepsilon(\varepsilon - 2^{2k_0+1-k}/l^2)$ and in time $2t + \mathcal{O}((\log N)^3)$ we can compute a set S with $m \in S$ and $\#S \leq l$.*

If we set $l = 1$ we get the following corollary:

Corollary 1 *Assume there is an adversary that on input $N, e, m^e \bmod N$ returns the $k - k_0$ most significant bits of m with advantage ε and in time t , where N is a k -bit RSA modulus, e is a public key belonging to N and $2k_0 < k$ holds. Then we can break the RSA problem related to (N, e) with advantage at least $\varepsilon(\varepsilon - 2^{2k_0+1-k})$ and in time $2t + \mathcal{O}((\log N)^3)$.*

Note that this achievement is the more valuable the smaller the difference $k - 2k_0$ is. However, in the case of PKCS #1 v2.0, k_0 is much smaller than $k/2$, therefore in this case the result is rather of theoretical nature.

4.2 Application to RSA-Paillier Cryptosystem

In section 2.2 we have described the reduction proof given by Catalano et al. [CNS02]. Remember that they have constructed the following congruence

$$Ax = y \bmod N^2, \quad A = a(1 + zN)^{-1} = a(1 - zN) \bmod N^2, \quad (8)$$

which is known to have the solution (r, μ) , where r, μ are elements of $\mathbb{Z}/N\mathbb{Z}$ and r is the sought-after RSA message. Hence (r, μ) is a small solution of (8) as described in subsection 3.2, where we have seen how to find all small solutions. To be concrete, `Lin.Cong` on the input $(A, N^2 - A, N^2, N)$ finds the smallest non-zero solution of (8) and all other small solutions come as integer multiples of this special solution.

We describe the explicit reduction algorithm as follows:

OW.RSA.Paillier

Input: (N, e) RSA Public-key, c ciphertext, \mathcal{O}_{RSAP} Hensel-Lifting oracle

Output: Message r such that $c = r^e \bmod N$ or an integer divisor of r

-
1. obtain $t = \mathcal{O}_{RSAP}(c)$
 2. generate random $a \in (\mathbb{Z}/N\mathbb{Z})^\times$
 3. obtain $s = \mathcal{O}_{RSAP}(a^e c \bmod N)$
 4. compute $v = ta^e s^{-1} \bmod N^2$
 5. compute $z = \frac{(v-1)}{N} e^{-1} \bmod N$
 6. compute $A = a(1 - zN) \bmod N^2$
 7. compute $(\hat{x}, \hat{y}) = \text{Lin.Cong}(A, N^2 - A, N^2, N)$
 8. return $\hat{x} + 1$
-

Obviously, the running time of this algorithm is $\mathcal{O}((\log N)^3)$ plus the time needed for calling the Hensel-Lifting oracle twice. To receive the original value r , we have to test if $(kr)^e = c \bmod N$, where the multiplier k runs from 1 to (the unknown number) $\gcd(r, \mu)$. In the following, we upper-bound the probability that $\gcd(r, \mu)$ is not sufficiently small. We exploit the following estimate (see [NZM91]):

$$\frac{\pi^2}{3} \left(\frac{2N^2 - 2N}{4N^2 + 4N + 1} \right) < \sum_{i=1}^N \frac{1}{i^2} < \frac{\pi^2}{3} \left(\frac{2N^2 + 2N}{4N^2 + 4N + 1} \right).$$

Hence we have

$$\begin{aligned} \#\{(a, b) \in [1, \dots, N]^2 \mid \gcd(a, b) > B\} &< \sum_{i=B+1}^N \frac{N^2}{i^2} \\ &< \frac{N^2 \pi^2}{3} \left(\frac{2N^2 + 2N}{4N^2 + 4N + 1} - \frac{2B^2 - 2B}{4B^2 + 4B + 1} \right) \\ &< \frac{N^2 \pi^2}{3} \left(\frac{1}{2} - \frac{2B^2 - 2B}{4B^2 + 4B + 1} \right). \end{aligned}$$

As a simple computation shows that

$$\frac{1}{2} - \frac{2B^2 - 2B}{4B^2 + 4B + 1} < \frac{1}{B},$$

we finally conclude

$$\#\{(a, b) \in [1, \dots, N]^2 \mid \gcd(a, b) > B\} < \frac{4N^2}{B}.$$

The values r and μ are independently chosen and uniformly distributed elements of $\mathbb{Z}/N\mathbb{Z}$. Replacing $5\varepsilon^{-2}$ for B , we therefore deduce that the probability that $\gcd(r, \mu)$ exceeds $5\varepsilon^{-2}$ is bounded above by $4\varepsilon^2/5$.

This leads to the following theorem:

Theorem 4. *Let \mathcal{O}_{RSAP} be the Hensel-lifting oracle that computes $r^e \bmod N^2$ for given $r^e \bmod N$ with advantage ε and in time t . Using \mathcal{O}_{RSAP} as a subroutine, we can break the RSA problem (N, e) with advantage $\varepsilon' > \varepsilon^2/5$ and in time $2t + \mathcal{O}((\log N)^3\varepsilon^{-2})$.*

An Example of OW_RSA_Paillier. We present a small example of reduction algorithm OW_RSA_Paillier. We choose the public-key of the RSA-Paillier cryptosystem as $(e, N) = (7, 9359629)$. In our case N^2 is equal to 87602655017641. Let $c = 2592708$ be the target ciphertext. We intend to find the integer r such that $c = r^e \bmod N$ using the oracle \mathcal{O}_{RSAP} .

In step 1 we ask c to oracle \mathcal{O}_{RSAP} , and we obtain $t = \mathcal{O}_{RSAP}(c) = 37278188147938$. In step 2, a random integer $a \in (\mathbb{Z}/N\mathbb{Z})^\times$ is generated, and we choose $a = 5973500$. In step 3, we compute $\mu^e = a^e c \bmod N$, ask it to oracle $\mathcal{O}_{RSAP}(\mu^e \bmod N)$, then we obtain $\mu^e \bmod N^2 = 59913274976876$. In step 4 and 5, integer z such that $ar = \mu(1 + zN) \bmod N^2$ is computed, and in our case $z = 9040417$. In step we obtain the linear equation $Ar = \mu \bmod N^2$ for $A = 35049167803493$ and two unknown variables $0 < r, \mu < N$.

In the following, we solve this linear equation using algorithm Lin_Cong. We list up the intermediate values of N', a', c' and y' , where N', a' and c' are initialized with N^2, A and $N^2 - A$. The while loop terminates if $y' < N$ holds. Step 10 and 11 of Lin_Cong are dedicated to compute the output values \hat{x} and \hat{y} , which in our case equal $r - 1$ and μ .

N'	a'	c'	y'	
87602655017641	35049167803493	52553487214148	35049167803493	
35049167803493	17544848392838	17504319410655	17544848392838	
17544848392838	40528982183	17504319410655	40528982183	
40528982183	4200892401	36328089782	4200892401	
4200892401	1479941827	2720950574	1479941827	
1479941827	238933080	1241008747	238933080	
238933080	192589733	46343347	192589733	
53559692	7216345	46343347	7216345	\leftarrow loop exit

The output values are $(1835097, 7216345) = (r - 1, \mu)$. Namely, we successfully find $r = 1835098$.

5 Conclusion

In this paper we investigated several security reduction algorithms related to RSA-OAEP and RSA-Paillier cryptosystems. These algorithms require to solve

a linear modular equation with a small solution. The standard algorithms for solving this task are Gaussian reduction and Euclidean reduction. We proposed an efficient alternative algorithm and showed its preferences. In the case of RSA-OAEP we were able to enhance the advantage of the reduction proof. Referring to RSA-Paillier, the use of our new algorithm provides us the complete security reduction proof, including explicit bounds for time costs and the achieved advantage.

References

- BR95. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption - how to encrypt with RSA. In *Advances in Cryptology - EUROCRYPT 84*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.
- CGHGN01. D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Nguyen. Paillier's cryptosystem revisited. In *The 8th ACM conference on Computer and Communication Security*, pages 206–214, 2001.
- CNS02. D. Catalano, P. Nguyen, and J. Stern. The hardness of Hensel lifting: The case of RSA and discrete logarithm. In *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 299–310, Berlin, 2002. Springer-Verlag.
- FOPS01. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274, 2001.
- GM97. M. Girault and J.-F. Misarsky. Selective forgery of RSA signatures using redundancy. In *Advances in Cryptology - EUROCRYPT 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 495–507. Springer-Verlag, 1997.
- HW79. G. Hardy and E. Wright. *An Introduction to The Theory Of Numbers*. Oxford Press, fifth edition edition, 1979.
- JC86. W. De Jonge and D. Chaum. Attacks on some RSA signatures. In *Advances in Cryptology - CRYPTO 85*, volume 218 of *Lecture Notes in Computer Science*, pages 18–27. Springer-Verlag, 1986.
- Kat01. S. Katzenbeisser. *Recent Advances in RSA Cryptography*. Kluwer Academic Publishers Group, 2001.
- MG02. D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems – A Cryptographic Perspective*. Kluwer Academic Publishers Group, 2002.
- NZM91. I. Niven, H.S. Zuckerman, and H.L. Montgomery. *An Introduction to the Theory of Numbers*. John Wiley & Sons, Inc., 1991.
- SF. C. P. Schnorr and R. Fischlin. Gittertheorie und algorithmische Geometrie. available from <http://ismi.math.uni-frankfurt.de/schnorr/lecturenotes/gitter.pdf>.
- Sho02. V. Shoup. OAEP reconsidered. *Journal of Cryptology*, (15):223–249, 2002.
- ST02. K. Sakurai and T. Takagi. New semantically secure public-key cryptosystems from the RSA primitive. In *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002.

A Proof of Lemma 1

Before starting the remaining proof of lemma 1, we recapitulate the notations given in section 3: We write (\hat{x}, \hat{y}) for the unique x -minimal solution of the congruence $ax = y + c \pmod{N}$ w.r.t. the bound B . The variables y_i, a_i, N_i and c_i constitute the corresponding values produced by the algorithm `Lin_Cong` (Outline) in the i th iteration of the while loop, and the value x_i is computed from $a_{i-1}, c_{i-1}, \hat{y}$ and N_{i-1} by $x_i = \frac{a_{i-1}x_{i-1} - \hat{y} - c_{i-1}}{N_{i-1}}$. The linear modular congruence $a_i x = y + c_i \pmod{N_i}$ is abbreviated by cong_i .

At first, we introduce a kind of “solution lifting”:

Proposition 1 *Let (x, y) be a solution of cong_{i+1} . Then the pair $\left(\frac{y+c_i+xN_i}{a_i}, y\right)$ is a solution of cong_i . If in addition $0 \leq x, y < N_{i+1}$ holds then $0 \leq \frac{y+c_i+xN_i}{a_i} \leq N_i$ is fulfilled.*

Proof. Note that the value $a_i = N_{i-1}$ cannot be zero, since otherwise iteration $(i+1)$ would not have been reached. First, we show $\frac{y+c_i+xN_i}{a_i} \in \mathbb{Z}$. The recursion formulas define $N_{i+1} = a_i$ and $c_{i+1} = c_i \pmod{N_{i+1}} = c_i \pmod{a_i}$. Hence there is an integer l such that c_{i+1} equals $c_i + la_i$. As (x, y) is a solution of cong_{i+1} we conclude

$$a_i \mid y + c_{i+1} - xa_{i+1} = y + (c_i + la_i) - x(-N_i \pmod{a_i}) \Rightarrow a_i \mid y + c_i + xN_i$$

It follows immediately that $\left(\frac{y+c_i+xN_i}{a_i}, y\right)$ is an integer solution of cong_i .

Now assume $0 \leq x, y < N_{i+1} = a_i$. We have

$$y + c_i + xN_i \leq y + c_i + (a_i - 1)N_i < a_i + N_i + (a_i - 1)N_i = a_i + a_iN_i$$

Therefore we conclude $\frac{y+c_i+xN_i}{a_i} < N_i + 1$, which finishes the proof.

Now we are prepared to prove lemma 1.

Lemma 1. *Let (x_i, \hat{y}) be the x -minimal solution of cong_i w.r.t. B and let $x_i > 0$. Then (x_{i+1}, \hat{y}) is the x -minimal solution of cong_{i+1} w.r.t. B . In particular, the y -value of the current x -minimal solution w.r.t. B does not change during the transformation $f_{\text{Lin_Cong}}$, as long as x_i is non-negative.*

Proof. Assume that $(x_{\text{alt}}, y_{\text{alt}})$ is a solution of cong_{i+1} where $0 \leq x_{\text{alt}} < N_{i+1}$ and $0 \leq y_{\text{alt}} < B$. Our goal is to show $x_{\text{alt}} \geq x_{i+1} \geq 0$.

First we prove that x_{i+1} is non-negative. Note that $a_i > 0$ must hold because otherwise the iteration $(i+1)$ of the while loop would not have been reached. From the definition of $x_{i+1} = \frac{a_i x_i - \hat{y} - c_i}{N_i}$ and the condition $x_i > 0$ we therefore conclude $\hat{y} + c_i + x_{i+1}N_i > 0$. Assume $x_{i+1} < 0$. Hence we have

$$\hat{y} + c_i > N_i \Rightarrow \hat{y} > N_i - c_i \geq -c_i \pmod{N_i} = y_i.$$

Thus $(0, y_i)$ is a solution of cong_i with $y_i < B$. This contradicts the preconditions, namely that (x_i, \hat{y}) is the x -minimal solution of cong_i w.r.t. B and $x_i > 0$. Consequently, we must have $x_{i+1} \geq 0$.

From proposition 1 we conclude that the pair

$$\left(\frac{y_{alt} + c_i + x_{alt}N_i}{a_i}, y_{alt} \right)$$

is a solution of cong_i , in particular we have $0 \leq \frac{y_{alt} + c_i + x_{alt}N_i}{a_i} \leq N_i$. As (x_i, \hat{y}) is the x -minimal solution of cong_i w.r.t. B , we conclude

$$\begin{aligned} \frac{y_{alt} + c_i + x_{alt}N_i}{a_i} - x_i &\geq 0 \\ \Rightarrow x_{alt} &\geq \frac{a_i x_i - y_{alt} - c_i}{N_i}. \end{aligned} \quad (9)$$

In the case of $y_{alt} \leq \hat{y}$ this immediately leads to the desired result $x_{alt} \geq x_{i+1} = \frac{a_i x_i - \hat{y} - c_i}{N_i}$. Thus we consider the case $y_{alt} > \hat{y}$. Looking at the difference between x_{i+1} and the right side of (9) we observe

$$x_{i+1} - \frac{a_i x_i - y_{alt} - c_i}{N_i} = \frac{-\hat{y} + y_{alt}}{N_i} < \frac{B}{N_i} < 1. \quad (10)$$

Note that the last inequality must hold since in the case of $N_i \leq B$ the iteration $(i + 1)$ of the while loop would not have been reached. Therefore we finally conclude $x_{alt} \geq x_{i+1}$ from (9), (10), and the fact that both of x_{alt} and x_{i+1} are integers.

The Insecurity of Esign in Practical Implementations

Pierre-Alain Fouque¹, Nick Howgrave-Graham²,
Gwenaëlle Martinet³, and Guillaume Poupard³

¹ École Normale Supérieure, Département d'Informatique
45 rue d'Ulm, 75230 Paris Cedex 05, France
`Pierre-Alain.Fouque@ens.fr`

² NTRU Cryptosystems, 5 Burlington Woods
Burlington, MA 02144 USA
`nhowgravegraham@ntru.com`

³ DCSSI Crypto Lab, 51, Boulevard de Latour-Maubourg
75700 Paris 07 SP, France

`Gwenaëlle.Martinet@worldonline.fr`, `Guillaume.Poupard@m4x.org`

Abstract. Provable security usually makes the assumption that a source of perfectly random and secret data is available. However, in practical applications, and especially when smart cards are used, random generators are often far from being perfect or may be monitored using probing or electromagnetic analysis. The consequence is the need of a careful evaluation of actual security when idealized random generators are implemented.

In this paper, we show that Esign signature scheme, like many cryptosystems, is highly vulnerable to so called partially known nonces attacks. Using a 1152-bit modulus, the generation of an Esign signature requires to draw at random a 768-bit integer. We show that the exposure of only 8 bits out of those 768 bits, for 57 signatures, is enough to recover the whole secret signature key in a few minutes.

It should be clear that we do not cryptanalyze a good implementation of Esign nor do we find a theoretical flaw. However, our results show that random data used to generate signatures must be very carefully produced and protected against any kind of exposure, even partial.

As an independent result, we show that the factorization problem is equivalent to the existence of an oracle returning the most or least significant bits of $S \bmod p$, on input S randomly chosen in \mathbb{Z}_{pq} .

Keywords: Esign signature scheme, Lattice reduction, LLL algorithm, Factorization problem.

1 Introduction

Most cryptographic systems make use of random sources for a range of applications. Random data may, for example, be transformed into secret or private keys for encryption or signature. From a provable security point of view, it is common to assume one has access to a source of perfect randomness. However, such an assumption is far from being totally realistic in many practical applications. The

first problem is that a true random number generator must be based on some kind of physical noise source. Such a generator is not commonly accessible on standard computers. When smart cards are used, the situation is even worse since such devices only have access to a very poor and constrained environment. The consequence is that random data is often simulated using a pseudo-random number generator.

In practical applications, there is a danger of adding weaknesses by using a biased generator or a weak pseudo-random number generator. Furthermore, with devices such as smart cards, the risk of secret data exposure by the way of probing or electromagnetic analysis may be increased if the random number generator is separated from the rest of the chip. As a consequence, a crucial question, when we consider practical security, is the impact of partial exposure of this random data for systems which have been proved secure under the assumption that a source of perfectly random and secret data is available.

The answer strongly depends on the application one considers. Usually, key generation is viewed as a crucial issue and people agree that a lot of care must be applied to the production of key material. However, does the exposure of one third of a 128-bit AES key have any real practical implication in usual applications? Such a question is of course rather controversial, but the complexity of an exhaustive search on the remaining secret bits, about 2^{85} block encryptions, might still be thought prohibitive. The same reasoning may be applied to other applications such as the choice of nonces or initial vectors. However, in some cases, partial exposure of secret information can have a far more dramatic consequence on the security of the system. Our first example is related to RSA with short public exponent. Boneh, Durfee and Frankel [6] have shown that the exposure of a quarter of the secret exponent enables one to factor the modulus in polynomial time. Similar results on DSA signature scheme are even more impressive. This scheme uses 160 bits of fresh random data, often called on-time key or ephemeral key, for each signature generation. It is well known that the exposure of those data enables to recover the secret signature key very easily. Howgrave-Graham and Smart [16] applied lattice reduction techniques to prove that the knowledge of only 8 bits out of the 160 bits of ephemeral keys for 30 signed messages enables to recover the secret key in a few seconds! In the same vein, following Boneh and Venkatesan [7], Nguyen and Shparlinski [19] have shown that indeed only 3 bits out of the 160 bits of each one-time key, for 100 messages, are enough to make the attack feasible. Finally, Bleichenbacher [3] has shown that if just one bit out of the 160 bits is biased, as was the case with the pseudo-random generator initially proposed by NIST [21], it is possible to mount an attack with time complexity 2^{64} , memory complexity 2^{40} and 2^{22} signatures.

Another analysis of the security of DSA in practical implementations, was done by Bellare, Goldwasser and Micciancio [2]. They did not assume partial exposure of ephemeral keys but their randomness was generated by a weak pseudo-random number generator, namely the linear congruential generator. In this case, DSA is totally insecure and the knowledge of a few signatures leads to the computation of the secret signature key.

All these results show that in some applications, such as DSA, data must be perfectly random and must remain completely secret. This does not mean that DSA is not secure but it points out a potential source of weakness. In actual implementations, the mechanism used to generate random data must be carefully chosen and evaluated, both from an algorithmic point of view and from a technological point of view. For example, electromagnetic analysis or probing techniques may enable one to learn a few random bits, even if it is not possible to recover the whole secret by these means. The above mentioned results show that the knowledge of a very small part of those bits is enough to totally break systems such as DSA.

Our Results

In this paper, we focus on the practical security of the Esign signature scheme [11]. Of course, in practical applications, this scheme is much less used than DSA. However, Esign could be preferred in many scenarios from a computational efficiency point of view. This is important when the signature device has low computing resources, which is the case with smart cards for instance. For applications such as on the fly signature with a contactless card (typical for fast and secure payment in the subway), Esign may be a very good candidate. Its practical security must consequently be carefully analyzed.

The technique we develop, and apply to careless Esign implementations, is of independent interest. It may be applied to other factorization based cryptosystems. Assuming partial exposure of a very small part of some secret data, our lattice reduction based technique allows one to factor the modulus very efficiently. Typically, this may be applied to the optimization of some SPA/DPA attacks on RSA systems [22,10].

In this paper, we describe an efficient technique based on the partial exposure of a few bits of Esign ephemeral keys. More precisely, using a 1152-bit modulus, the generation of an Esign signature requires to draw at random a 768-bit integer. We show that the exposure of only 8 bits out of those 768 bits for 57 signatures is enough to recover the all secret signature key in a few minutes.

It should be clear that we do not propose neither a cryptanalysis of Esign nor a theoretical flaw. However, our results show that random data used to generate signatures must be very carefully produced and protected against any kind of exposure, even partial.

Previous Works

The *hidden number problem* (HNP) has been described by Boneh and Venkatesan in [7] in order to prove the hardness of the most significant bits of secret keys in Diffie-Hellman and related schemes in prime fields. The HNP can be defined as follows: given s_1, \dots, s_d chosen uniformly and independently at random in \mathbb{Z}_q^* and $\text{MSB}_\ell(\alpha s_i \bmod q)$ for all i , the problem is to recover $\alpha \in \mathbb{Z}_q^*$. Here, $\text{MSB}_\ell(x)$ for $x \in \mathbb{Z}_q$ denotes any integer z satisfying $|x - z| < q/2^{\ell+1}$. In [7], the authors present a simple solution to this problem by reducing HNP

to a lattice closest vector problem (CVP). In particular, they show that the HNP can be solved if $\ell \geq \sqrt{\log(q)} + \log(\log(q))$ and $d = 2\sqrt{\log(q)}$. According to [20], using the best known polynomial-time CVP approximation algorithm due to Ajtai *et al.* [1] and Kannan [17], ℓ can be asymptotically improved to $O(\sqrt{\log(q)} \log(\log(\log(q)))) / \log(\log(q))$.

In this paper we consider a problem related to a HNP problem modulo a *secret* value and we propose an algorithm to solve it. In [4], Boneh also mentions the HNP modulo $N = pq$. Now, p and q denote the factors of a modulus N . Our problem can be formulated as follows: given s_1, \dots, s_d chosen uniformly and independently at random in \mathbb{Z}_N^* and $\text{MSB}_\ell(s_i \bmod p)$ for all i , the problem is to recover p . Our algorithm uses the *orthogonal lattice theory* of [20] to obtain several small lattice vectors. Moreover, we also use the extension of Nguyen and Shparlinski [19] if the distribution of the s_i is not necessarily perfectly uniform using the discrepancy notion. Indeed, if we note $|s|_p = \min_{b \in \mathbb{Z}} |s - bp|$ for any integer s , the values s_i in the lattice are such that $|s_i|_p < p/2^\ell$ and are thus not uniformly distributed in \mathbb{Z}_p . If N is a 1024-bit modulus, then the results of the HNP say that with $d = 64$ and $\ell = 9$, N can be factored. We get similar results with our algorithm.

Finally, contrary to the lattice based algorithm used by Boneh, Durfee and Howgrave-Graham [5], our factorization algorithm uses an oracle. In some cases, this oracle can be found in practical implementations. For example, if the pseudorandom generator of the nonces used in Esign implementation is biased such that the MSBs can be learned, then we can break the signature scheme by factoring the modulus. In this application, the secret modulus is a composite number pq and $N = p^2q$. This paper can be seen as an extension of previous attacks on signature schemes, based on the discrete log such as DSA in [19,16], to some factorization based signature schemes.

The results in this paper were independently discovered, but are of a similar vein to those found in the Esign technical review [15].

2 Description of Esign

Esign is a signature scheme proposed by Okamoto and Shiraishi in 1985 [23]. It is based on modular computations with special form modulus. The main advantage of Esign is its efficiency. Compared to RSA or EC based scheme, Esign is several times faster in terms of signature and verification performance.

Let $N = p^2q$ a $3k$ -bit integer, with p and q two primes of roughly the same length. The secret key consists in the two k -bit primes p and q . The public key is (N, e) , where e is an integer larger than 4. The scheme uses a cryptographic hash function H to compute $(k-1)$ -bit long message digests. The signature of a message M is performed as follows:

1. the message M is first hashed into $H(M)$. We denote by y the integer corresponding to the $3k$ -bit string $0\|H(M)\|0^{2k}$, where 0^{2k} denotes the concatenation of $2k$ null bits,
2. An integer r is randomly chosen in \mathbb{Z}_{pq}^* ,

3. Compute:

- (a) $z = y - r^e \bmod N$,
- (b) $w_0 = \left\lceil \frac{z}{pq} \right\rceil$,
- (c) $w_1 = w_0 pq - z$. If $w_1 > 2^{2k-1}$, then come back to step 2,
- (d) $u = w_0(er^{e-1})^{-1} \bmod p$,
- (e) $s = r + upq$,

4. Return s as a signature for M .

Note that in the rest of this paper, we often write signatures as the sum of the random nonce r and a multiple $u \times pq$ of the secret key pq .

To verify if a signature s is valid for the message M , a verifier simply checks if the k most significant bits of $s^e \bmod N$ are equal to $0\|H(M)$. The verification algorithm is consistent since:

$$\begin{aligned} s^e &= (r + upq)^e \bmod N \\ &= r^e + er^{e-1}upq \bmod N \\ &= (y - z) + w_0pq \bmod N \\ &= y + w_1 \bmod N \end{aligned}$$

Since $w_1 < 2^{2k-1}$, and N is exactly $3k$ bits long, the k most significant bits of $s^e \bmod N$ are those of y , i.e. $0\|H(M)$.

The security of Esign is based on a variant of the RSA problem which consists in computing modular e -th roots. More precisely, even the computation of approximations of such roots seems to be difficult. The Approximate e -th Root (AER) problem is formally defined as follows:

Given a modulus $N = p^2q$, an exponent $e \geq 4$ and $y \in \mathbb{Z}_N^$, find $x \in \mathbb{Z}_N^*$ such that $x^e \in [y, y + 2^{2k-1}]$.*

The knowledge of the factorization of N gives an efficient solution to this problem. Without p or q , this problem is supposed to be hard. The AER assumption is that the AER problem is intractable.

The initial scheme proposed in [23] was based on the exponent $e = 2$. This version has been cryptanalyzed the same year by Brickell and DeLaurentis in [8]. The cubic scheme has also been broken using lattice reduction (see in particular [9,13,26]). However, for $e \geq 4$, no attack has been reported for the moment. A potential way to break the signature scheme is to factor the modulus N and then to recover the secret key. This constitutes a total break of the scheme. Note that if the random value r is compromised for just one signature, the factorization can be easily recovered. Indeed, since $s = r + upq$, if r is known, then the GCD of the modulus $N = p^2q$ and $s - r$ reveals pq .

We also notice that the knowledge of $r^e \bmod N$ allows to recover the prime factors p and q . Indeed, s^e can be written as $r^e - er^{e-1}upq \bmod N$ and the GCD of N and $s^e - r^e \bmod N$ gives pq . The secrecy of the random values is consequently a crucial issue for Esign.

Moreover, the scheme with $e \geq 4$ and e prime with $\phi(N) = p(p-1)(q-1)$, is provably secure in the random oracle model. More precisely, it is proved secure against existential forgeries in Single Occurrence Chosen Message Attacks scenarios, under the AER assumption (see [25]). An adversary querying a signature oracle for messages of his choice, but with the restriction that a message cannot be submitted twice to the oracle, cannot forge a signature for a message. Otherwise, he can solve the AER problem, supposed to be intractable. Extending the proof to the stronger adaptive chosen message attacks model is an open problem. Thus, two different ways have been proposed to make Esign provably secure in the strong sense [14]. The first method, called Esign-D, is deterministic: the random nonce r is generated from the message to sign and an additional secret string, included in the private key. The second one, called Esign-R, uses another random nonce ρ , given as part of the signature, to generate the hash of the message as $H(M\|\rho)$. In the following, the attacks we present are not chosen message ones, but are based on flawed implementations. Hence, they do not depend on the version used. So without loss of generality, we focus on the first scheme described above.

3 Lattice Based Attacks

In this section we first recall some basic facts about lattices and reduction algorithms. Then, we detail how to use lattice reduction in order to factor modulus such as N under some assumptions on the random data used in Esign.

3.1 Lattice Reduction

Notations. Let $N = p^2q$ an Esign modulus. Then any integer s in \mathbb{Z}_N can be written as $s = r + upq$ with $0 \leq r < pq$ and $0 \leq u < p$.

Definitions. In the following, we denote by $\|\mathbf{x}\|$ the Euclidean norm of the vector $\mathbf{x} = (x_1, \dots, x_{d+1})$, defined by $\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{d+1} x_i^2}$. Let $\mathbf{v}_1, \dots, \mathbf{v}_d$, be d linearly independent vectors such that for $1 \leq i \leq d$, $\mathbf{v}_i \in \mathbb{Z}^{d+1}$. We denote by L , the lattice spanned by the matrix V whose rows are $\mathbf{v}_1, \dots, \mathbf{v}_d$. L is the set of all integer linear combinations of $\mathbf{v}_1, \dots, \mathbf{v}_d$:

$$L = \left\{ \sum_{i=1}^d c_i \mathbf{v}_i, c_i \in \mathbb{Z} \right\}$$

Geometrically, $\det(L) = \det(V \times V^T)$ is the volume of the parallelepiped spanned by $\mathbf{v}_1, \dots, \mathbf{v}_d$. The Hadamard's inequality says that $\det(L) \leq \|\mathbf{v}_1\| \times \dots \times \|\mathbf{v}_d\|$.

Given $\langle \mathbf{v}_1, \dots, \mathbf{v}_d \rangle$ the LLL algorithm [18] will produce a so called "reduced" basis $\langle \mathbf{b}_1, \dots, \mathbf{b}_d \rangle$ of L such that

$$\|\mathbf{b}_1\| \leq 2^{(d-1)/2} \det(L)^{1/d} \quad (1)$$

in time $O(d^4 \log(M))$ where $M = \max_{1 \leq i \leq d} \|\mathbf{v}_i\|$. Consequently, given a basis of a lattice, the LLL algorithm finds a short vector \mathbf{b}_1 of L satisfying equation (1). Moreover, we assume in the following that the new basis vectors are of the same length and also have all their coordinates of approximatively the same length. Indeed, a basis for a random lattice can be reduced into an almost orthonormal basis. Therefore, $\|b_i\| \approx \|b_1\|$ for $1 \leq i \leq d$, and so $\|b_i\|^d \approx \det(L)$.

3.2 Lattice-Based Factoring Algorithm

In this subsection, we present a lattice technique to factor a modulus $N = p^2q$, where p and q are two k -bit primes, given an oracle $\mathcal{O}_{\ell,pq}$ that, on input $\tilde{s} \in \mathbb{Z}_N$, returns the ℓ MSBs of $\tilde{s} \bmod pq$. We will see in section 4 that in practical applications it is sometimes possible to realize such an oracle. In the following we denote by $n = 3k$ the bit length of N .

Let $\tilde{s} \in \mathbb{Z}_N$ be an integer smaller than N . If an $\mathcal{O}_{\ell,pq}$ oracle is available, let us query the ℓ most significant bits of $\tilde{s} \bmod pq$; we denote by t the answer of the oracle. Then, $s = \tilde{s} - t \times 2^{2k-\ell}$ may always be written as $r + upq$ with $0 \leq r < pq/2^\ell$ and $0 \leq u < p$. Finally, after d queries to the oracle, we may consider that we know d integers $s_i \in \mathbb{Z}_N$ such that $s_i = r_i + u_i pq$ with $0 \leq r_i < pq/2^\ell$ and $0 \leq u_i < p$. However, the r_i and u_i values are unknown. Our objective is to recover pq .

First we note that if we are able to recover one of the u_i , then recovering the factors p and q of N can be efficiently done. Indeed, we suppose first that the recovered u_i value is larger than $p/2^\ell$. This occurs with probability $1 - 1/2^\ell$ and if this is not true, we can recover another u_i until this event occurs. Thus, we have $p/2^\ell < u_i < p$ and we can write $s_i/u_i = r_i/u_i + pq$ where r_i/u_i is at most k bits. Consequently, the k most significant bits of s_i/u_i are those of pq . We denote by A the integer matching pq on its k MSBs and zeroing the k least significant bits. The $2k$ -bit value A is known and we can write $pq = A + \alpha$ where $\alpha < 2^k$ is unknown. Finally, since $N = p \times pq = p \times (A + \alpha)$, we have:

$$\frac{N}{A} = p + \frac{p\alpha}{A} \quad (2)$$

where $0 \leq \frac{p\alpha}{A} < 2$, since $p\alpha$ is at most of $2k$ bits and A is exactly $2k$ bits. Thus, p equals either $\lfloor N/A \rfloor$ or $\lfloor N/A \rfloor - 1$.

In the following, we present an algorithm to recover all the u_i . In a first phase, the algorithm looks for small linear integer combinations of the s_i using the LLL algorithm. Then, in a second phase, we solve a linear system to recover the u_i . In the sequel, we describe these two phases.

Finding Small Linear Integer Combinations of the u_i . The following lemma shows that searching a small linear integer combination of the s_i with small coefficients is sufficient to find a null linear combination of the u_i .

Lemma 1. *Let $N = p^2q$ be a n -bit modulus with p and q of roughly the same length. Let s_1, \dots, s_d be d random integers in \mathbb{Z}_N , $s_i = r_i + u_i pq$ such that $|r_i| < pq/2^\ell$.*

If there exist d integers c_i , for $1 \leq i \leq d$, such that $c = \max_{1 \leq i \leq d} |c_i| < 2^\ell/d$ and $|\sum_{i=1}^d c_i s_i| < pq$, then $\sum_{i=1}^d c_i u_i \in \{-1, 0, 1\}$.

Moreover, if $c < 2^\ell/2d$ and $|\sum_{i=1}^d c_i s_i| < pq/2$, then $\sum_{i=1}^d c_i u_i = 0$.

Proof. By definition, we have $\sum_{i=1}^d c_i s_i = \sum_{i=1}^d c_i r_i + pq \sum_{i=1}^d c_i u_i$. Thus by the triangle inequality, we can write:

$$pq \left| \sum_{i=1}^d c_i u_i \right| \leq \left| \sum_{i=1}^d c_i s_i \right| + \left| \sum_{i=1}^d c_i r_i \right| \quad (3)$$

Moreover, since $c < 2^\ell/d$ and $|r_i| < pq/2^\ell$ for $1 \leq i \leq d$, then

$$\left| \sum_{i=1}^d c_i r_i \right| \leq \sum_{i=1}^d |c_i r_i| \leq d \times \left(\frac{2^\ell}{d} \times \frac{pq}{2^\ell} \right) \leq pq$$

Now we know that $|\sum_{i=1}^d c_i s_i| < pq$. Then from equation (3), $pq |\sum_{i=1}^d c_i u_i| < 2pq$ and $|\sum_{i=1}^d c_i u_i| < 2$. This proves the first part of the lemma. The second part of the lemma can be easily deduced from the previous computations. \square

Therefore, we look for small integer linear combination of the s_i , i.e. such that $|\sum_{i=1}^d c_i s_i| \leq pq$ and $c < 2^\ell/d$. From previous lemma, finding such a combination gives a linear equation in the u_i variables.

Now we present a lattice-based method to recover the coefficients of a small combination of the s_i . Suppose K is an integer less than N , whose exact value will be defined later. We consider the following $d \times (d+1)$ -matrix:

$$M = \begin{pmatrix} s_1 & K & 0 & \dots & 0 \\ s_2 & 0 & K & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ s_d & 0 & \dots & 0 & K \end{pmatrix}$$

The size of the original basis vector is approximately N since the s_i are integers in \mathbb{Z}_N . In order to estimate the size of a small vector returned by LLL, we upper bound the volume of the lattice L , spanned by the rows of M . In the following, we upperbound the determinant of the lattice L and show that

$$\det(L)^2 = K^{2d-2} (K^2 + \sum_{j=1}^d s_j^2)$$

Since L is not a full lattice, its volume is the square root of the determinant of the Gramian matrix [12], $M \times M^T$. Thus, we have:

$$\det(L)^2 = \det(M \times M^T) = \begin{vmatrix} s_1^2 + K^2 & s_1 \times s_2 & s_1 \times s_3 & \dots & s_1 \times s_d \\ s_2 \times s_1 & s_2^2 + K^2 & s_2 \times s_3 & \dots & s_2 \times s_d \\ s_3 \times s_1 & s_3 \times s_2 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & s_{d-1} \times s_d \\ s_d \times s_1 & \dots & \dots & s_d \times s_{d-1} & s_d^2 + K^2 \end{vmatrix}$$

We can factor the first row by s_1 , the second by s_2 , ..., s_d and similarly for the columns. Therefore the determinant can be written as

$$\det(L)^2 = \prod_{i=1}^d s_i^2 \times \begin{vmatrix} 1 + K^2/s_1^2 & 1 & 1 & \dots & 1 \\ 1 & 1 + K^2/s_2^2 & 1 & \dots & 1 \\ 1 & 1 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 1 \\ 1 & \dots & \dots & 1 & 1 + K^2/s_d^2 \end{vmatrix}$$

The last determinant can be computed exactly and is equal to

$$\prod_{i=1}^d \frac{K^2}{s_i^2} + \sum_{i=1}^d \prod_{j=1, j \neq i}^d \frac{K^2}{s_j^2}$$

and consequently,

$$\det(L)^2 = K^{2d-2} (K^2 + \sum_{j=1}^d s_j^2)$$

Therefore, since for all i , $|s_i| \leq N$ and $K \leq N$, the size of the small vector returned by LLL on this lattice is less than

$$2^{(d-1)/2} \times (d+1)^{1/2d} \times K^{\frac{d-1}{d}} N^{\frac{1}{d}}$$

For the present discussion we ignore factors like $2^{(d-1)/2}$ dependent only on the size of the matrix. Indeed, in practice, LLL returns a short vector much smaller than theoretical upperbounds. Consequently, we can assume that the shortest vector returned by LLL is of length about $(d+1)^{1/2d} \times K^{\frac{d-1}{d}} N^{\frac{1}{d}}$.

Now we fix $K = \left\lceil N^{\frac{2}{3} - \frac{1}{3(d-1)}} / 2 \right\rceil$. As a consequence, a simple computation shows that, in this case, the length of a short vector returned by LLL is less than

$$(d+1)^{1/2d} \times N^{\frac{2}{3} - \frac{1}{3d}}$$

which is less than pq since $\sqrt{d+1} \ll N^{1/3}$. Therefore a short vector has all its coordinates smaller than pq .

In the following, we show how a short vector \mathbf{b}_1 returned by LLL allows us to determine the coefficients of a short linear combination of the s_i . Due to the form of the matrix M , \mathbf{b}_1 can be written as

$$\mathbf{b}_1 = \left(\sum_{i=1}^d c_i \cdot s_i, K \cdot c_1, \dots, K \cdot c_d \right) \quad (4)$$

where the c_i are integers. We denote by c the maximum of the $|c_i|$. If \mathbf{b}_1 is a short vector returned by LLL, then all its coordinates are smaller than pq . In particular, we have $Kc < pq$. Consequently, $c < 2pq/N^{\frac{2}{3} - \frac{1}{3(d-1)}}$. Furthermore, if $\ell > \frac{n}{3(d-1)} + \log(d) + 1$, then

$$c < \frac{2N^{2/3}}{N^{\frac{2}{3} - \frac{1}{3(d-1)}}} \leq 2N^{\frac{1}{3(d-1)}} < \frac{2^\ell}{d}$$

Therefore, since $\sum_{i=1}^d c_i \cdot s_i < pq$ and $c < 2^\ell/d$, then lemma 1 implies that $\sum_{i=1}^d c_i u_i \in \{-1, 0, 1\}$.

As a consequence, if we have d random values $s_i = r_i + u_i pq$, where $|r_i| < pq/2^\ell$ and $\ell > \left\lceil \frac{n}{3(d-1)} + \log(d) + 1 \right\rceil$, then the shortest vector returned by LLL gives us the coefficients of a very small combination of the u_i and we finally have a linear equation in the u_i variables.

However, one equation is not sufficient to recover at least one u_i . In the second phase of our algorithm, we show that in fact we can obtain d very small linear combinations of the u_i .

Recovering the u_i . The vectors of the new lattice basis have the property to be all of about the same length. Consequently, each vector \mathbf{b}_i of the new basis gives a small integer combination of the s_i and so of the u_i . Experimentally, we observe that the linear combination of the u_i is null except for the last one which is equal to ± 1 .

Thus, each short vectors returned by LLL gives a small linear combination of the s_i . The matrix returned by the LLL algorithm can be expressed as $C \times M$ where

$$C = \begin{pmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{j,1} & c_{j,2} & \dots & c_{j,d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d,1} & c_{d,2} & \dots & c_{d,d} \end{pmatrix}$$

Each row of C contains the coefficients of a small linear combination of the s_i . The matrix C is invertible since its determinant is ± 1 and thus solving the system $\mathbf{u} \cdot C^T = (0, \dots, 0, 1)$, where $\mathbf{u} = (u_1, \dots, u_d)$ allows to recover the u_i .

Once the u_i are obtained, recovering half of the bits of pq is easy by computing $\lfloor \frac{s_i}{u_i} \rfloor$ for one of the value s_i . Then p is computed according to equation 2. Finally, we have the following theorem:

Theorem 1. *Let $N = p^2q$ be a n -bit modulus. Given an oracle $\mathcal{O}_{\ell,pq}$ that on input $s \in \mathbb{Z}_N$, returns the ℓ most significant bits of $s \bmod pq$ where $\ell \geq \left\lceil \frac{n}{3(d-1)} + \log(d) + 1 \right\rceil$ and $d < n$, there exists a probabilistic polynomial-time algorithm in n to factor N from d random and independent numbers s in \mathbb{Z}_N .*

3.3 Extending the Attack to the Least Significant Bits

In this paper, we focus on the importance of MSBs confidentiality. However, such a presentation has been chosen for simplicity reasons since the same analysis can be done with the least significant bits. More precisely the knowledge of the ℓ least significant bits of $S_i \bmod pq$, for d values $S_i \in \mathbb{Z}_N$, also allows us to factor N for the obvious reason that the knowledge of the least significant bits can be reduced to the knowledge of the most significant bits, as explained below.

Consider a $3k$ -bit Esign modulus $N = p^2q$. A value S randomly chosen in \mathbb{Z}_N can always be written as $S = r + upq$ where $0 \leq r < pq$ and $u < p$. Assume now that the ℓ LSBs of r , denoted by r_0 , are known. Then, the ℓ LSBs of $S - r_0 \bmod pq = r - r_0$ are zero. We now denote by r_1 the $(2k - \ell)$ -bit value $(r - r_0)/2^\ell$. Let a be the inverse of $2^\ell \bmod N$. We can note that a is also the inverse of 2^ℓ modulo pq . Consequently, we can compute

$$\begin{aligned} a \times (S - r_0) &= a \times (r - r_0) \bmod pq \\ &= a \times 2^\ell \times \frac{(r - r_0)}{2^\ell} \bmod pq \\ &= (1 \bmod pq) \times r_1 \bmod pq \\ &= r_1 \bmod pq \end{aligned}$$

Therefore, $s = a(S - r_0) \bmod N$ can be written as $r_1 + u_1pq$ for $u_1 < p$ and $r_1 < pq/2^\ell$. Thus s is a candidate input for the matrix M of the algorithm of the previous subsection.

3.4 Application to RSA Modulus

It is worth noticing that this algorithm is independent of the special form $N = p^2q$ of the modulus. It also works for any RSA modulus $N = pq$ as soon as:

$$\ell \geq \left\lceil \frac{n}{2(d-1)} + \log(d) + 1 \right\rceil$$

If $s_i \in \mathbb{Z}_N$ is written as $s_i = r_i + u_i p$, for $r_i < p/2^\ell$ and $u_i < q$, then we can recover the u_i and computing p from $\lfloor \frac{s_i}{u_i} \rfloor$.

As a consequence, if there exists an oracle $\mathcal{O}_{\ell,p}$ which on input $S \in \mathbb{Z}_N$ returns the ℓ most significant bits of $S \bmod p$ where p is a factor of the modulus

N , then we can factor N in polynomial time in $\log(N)$. Therefore the problem of finding the ℓ MSBs of $S \bmod p$ for d different random and independent values $S \in \mathbb{Z}_N$, is equivalent to the factorization problem.

4 Partially Known Nonces in Esign Signature Scheme

In the following we describe some potential flaws in practical implementations of Esign. The main idea is to notice that the secrecy and the randomness of all the nonces is a crucial security point: the knowledge of only a few bits of these random values is enough to efficiently recover the secret signature key.

Let $N = p^2q$ an Esign modulus where p and q are two k -bit primes such that $q < p$. The signature scheme is fully described in section 2.

We first consider an attack on Esign when the random nonces are not full-size. Suppose the random number generator is biased so that the most significant bits of the random values are always zero. We show how to efficiently factor the modulus from a small set of signatures by using the technique described in section 3.2. Precisely, suppose the random number generator produces nonces smaller than $pq/2^\ell$, for an integer $\ell \geq 1$, instead of randomly drawing uniformly distributed integers in the interval $[0, pq[$. We know that all the generated signatures may be written as $s = r + upq$ where r is the random nonce. Thus, a signature is a noisy multiple of the secret factor pq . If the number ℓ of null most significant bits of r is sufficiently large, then we can factor N by recovering p with the technique presented above. The attack goes as follows: suppose we have a set of d Esign signatures s_i , for any messages. Each can be written as $s_i = r_i + u_i pq$, for $1 \leq i \leq d$, and where $r_i \leq pq/2^\ell$ and $u_i < p$. As shown in section 3.2 we can recover the u_i by reducing a lattice with the LLL algorithm. As soon as we have $\ell \geq \left\lceil \frac{n}{3(d-1)} + \log(d) + 1 \right\rceil$, where n is the bit size of the modulus N . Then we can write:

$$\frac{s_i}{u_i} = \frac{r_i}{u_i} + pq$$

We remark that $\left\lfloor \frac{r_i}{u_i} \right\rfloor$ is at most a k -bit integer. Thus, we can finally recover p according to equation 2. Experimented results are provided below. The tests have been run on an Intel Pentium IV, XEON 1.5 GHz, with the Shoup's library NTL ([24]). For each modulus length n , we give the length of pq (that is also the expected length for the random r), the effective length of the nonce r , the experimental and theoretical bounds for ℓ , and the time needed to recover p and q . The number of required signatures is d .

We observe that the experimental bound for ℓ is better than expected. This can be simply explained by the good performances of the LLL algorithm. In practice, this algorithm works indeed better than expected and the vectors returned are shorter than the provable upper bounds. Another explanation can be made for this fact: in section 3.2, we have used a theoretical bound on the sum $\sum_{i=1}^d c_i s_i \leq dcN$. This bound has then been used to find the theoretical bound on ℓ . However, in practice, the sum $\sum_{i=1}^d c_i s_i$ is approximately $\sqrt{d} \cdot cN$ on

$n = 3k$	$2k$	$\log(r)$	experimental value for ℓ	theoretical bound for ℓ	d	time to factor
512	340	335	5	8	55	2 min 10
768	512	506	6	9	55	2 min 20
1024	682	674	8	11	56	2 min 30
1152	768	760	8	11	57	3 min
1536	1024	1013	11	14	57	4 min 10
2048	1364	1349	15	17	57	5 min 50

Fig. 1. Experimental results on Esign with partially known nonces.

average. Thus, this gives a smaller bound on ℓ : the algorithm works as soon as $\ell \geq \left\lceil \frac{n}{3(d-1)} + \frac{\log(d)}{2} + 1 \right\rceil$. This gives results closer to the experimental results. This bound is given in figure 1.

Hence, if the random number generator produces nonces in an interval smaller than expected, then recovering the secret key can be made from a small set of signatures, for any messages. However, even if the random values are generated in all the interval $[0, pq[$, the difference between two consecutive nonces should not be too small. Indeed, in this case, the same attack applies: considering the differences $s_{i+1} - s_i$ whose most significant bits modulo pq are small, gives the same results.

Thus, the random number generator is a crucial security point and the nonces should be generated uniformly and independently in the range $[0, pq[$. If we now consider physical attacks on probing or electromagnetic analysis, the attack can also be mounted as soon as the observation of the ℓ MSB or LSB of the random nonces is feasible. This may be realistic using smart cards.

5 Other Potential Weaknesses in Esign Implementations

In [2], Bellare, Goldwasser and Micciancio have pointed out that using linear congruential generator in DSS signature scheme is totally insecure. The secret key can be easily recovered in this case, and even if the outputs of the generator are truncated. As for DSS, using a linear congruential generator (LCG) with public parameters leads to insecure implementations of Esign.

Such a generator is parameterized by integers a, b, M and is based on a linear recurrence: $r_{i+1} = ar_i + b \bmod M$. The initial seed r_0 is the secret. We consider the security of Esign in this case and we show that the knowledge of only two signatures allows to recover the secret signature key. Suppose that Esign is used with the pseudo-random generator defined by $r_{i+1} = ar_i + b \bmod M$ where M is a secret multiple of pq , less than N , and a and b are public integers in \mathbb{Z}_M . The initial state r_0 , that should not be reset, is kept secret as part of the private key. The modulus M is chosen to be a multiple of pq so that after reduction modulo

pq in the signature generation, the generated random values are still uniformly distributed in the range $[0, pq[$. Such a choice seems to be the most natural one.

For any positive index i , we have $s_i = r_i + u_i pq$. Thus the following equality holds:

$$s_{i+1} = r_{i+1} + u_{i+1}pq = ((ar_i + b) \bmod M) \bmod pq + u_{i+1}pq$$

Thus, since a and b are public and M is a multiple of pq , one can compute $s_{i+1} - as_i - b$ which is a multiple of pq . Its GCD with the modulus N is pq , and the secret key is found.

Note that this can also be applied even if the parameter b is secret. With only four signatures s_i , s_{i+1} , s_j and s_{j+1} , the secret factor pq can also be recovered. Indeed, it suffices to compute $(s_{i-1} - s_i) - (s_{j+1} - s_j) = (u_{i+1} - u_i + u_{j+1} - u_j + K)pq$ where K is an integer. The GCD of N with this difference reveals pq .

Finally, using a linear congruential generator is insecure in this case.

6 Conclusion

In conclusion we have shown in this paper that Esign must be carefully implemented since like many other public key cryptosystems, security of ephemeral keys is of crucial importance. We also insist on the idea that physical techniques like probing or electromagnetic analysis can be very efficiently combined with more theoretical algorithmic cryptanalysis methods, for example based on LLL.

References

1. M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the 33rd Annual Symposium on the Theory of Computing (STOC) 2001*, pages 601 – 610. ACM Press, 2001.
2. M. Bellare, S. Goldwasser, and D. Miccianco. "Pseudo-Random" Number Generation within Cryptographic Algorithms: the DSS Case. In B. Kaliski, editor, *Advances in Cryptology – Crypto'97*, volume 1294 of *LNCS*, pages 277 – 291. Springer-Verlag, 1997.
3. D. Bleichenbacher. On the Generation of DSA One-Time Keys. In *The 6th Workshop on Elliptic Curve Cryptography (ECC 2002)*, 2002.
4. D. Boneh. Simplified OAEP for the RSA and Rabin Functions. In M. Bellare, editor, *Advances in Cryptology – Crypto'01*, volume 2139 of *LNCS*, pages 275 – 291. Springer-Verlag, 2001.
5. D. Boneh, G. Durfee, and N. Howgrave-Graham. Factoring $n = p^r q$ for large r . In M. Wiener, editor, *Advances in Cryptology – Crypto'99*, volume 1666 of *LNCS*, pages 326 – 337. Springer-Verlag, 1999.
6. D. Boneh, G. Durfee, and Y. Frankel. An attack on RSA given a fraction on the private key bits. In K. Ohta and D. Pei, editors, *Advances in Cryptology – Asiacrypt'98*, volume 1514 of *LNCS*, pages 25 – 34. Springer-Verlag, 1998.
7. D. Boneh and R. Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes. In N. Koblitz, editor, *Advances in Cryptology – Crypto'96*, volume 1109 of *LNCS*, pages 129 – 142. Springer-Verlag, 1996.

8. E. F. Brickell and J. M. DeLaurentis. An attack on a signature scheme proposed by Okamoto and Shiraishi. In H. C. Williams, editor, *Advances in Cryptology – Crypto ’85*, volume 218 of *LNCS*, pages 28 – 32. Springer Verlag, 1985.
9. E. F. Brickell and A. M. Odlyzko. Cryptanalysis: A Survey of Recent Results. In G. J. Simmons, editor, *Contemporary Cryptology – The Science of Information Integrity*, pages 501 – 540. IEEE Press, 1991.
10. P. A. Fouque, G. Martinet, and G. Poupard. Attacking Unbalanced RSA-CRT using SPA, 2003. To appear in the Proceedings of CHES’03.
11. E. Fujisaki, T. Kobayashi, H. Morita, H. Oguro, T. Okamoto, and S. Okasaki. ESIGN: Efficient Digital Signature Scheme, submission to NESSIE, 2000.
12. J. van zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.
13. M. Girault, P. Toffin, and B. Vallée. Computation of Approximate l -th Roots Modulo n and Application to Cryptography. In S. Goldwasser, editor, *Advances in Cryptology – Crypto’88*, volume 403 of *LNCS*, pages 100 – 117. Springer Verlag, 1988.
14. L. Granboulan. How to repair Esign. In *SCN’02*. Springer-Verlag, 2002.
15. N. Howgrave-Graham. A Review of the ESIGN digital signature standard, 2001. Available at http://www.shiba.tao.go.jp/kenkyu/CRYPTREC/fy15/cryptrec20030424_outrep.html. Report #1007.
16. N. Howgrave-Graham and N. P. Smart. Lattice Attacks on Digital Signature Schemes. *Design, Codes and Cryptography*, 23:283 – 290, 2001.
17. R. Kannan. Algorithmic geometry of numbers. *Annual Review of Computer Science*, 2:231 – 267, 1987.
18. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261(4):515 – 534, 1982.
19. P. Q. Nguyen and I. E. Shparlinski. The Insecurity of the Digital Signature Algorithm with Partially Known Nonces. *Journal of Cryptology*, 15(3):151 – 176, 2002.
20. P. Q. Nguyen and J. Stern. The Two Faces of Lattices in Cryptography. In J. Silverman, editor, *Proc. of Cryptography and Lattices Conference*, volume 2146 of *LNCS*, pages 146 – 180. Springer-Verlag, 2001.
21. NIST. Digital Signature Standard (DSS). Federal Information Processing Standards PUBLication 186, November 1994.
22. R. Novak. SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation. In D. Naccache and P. Paillier, editors, *Public Key Cryptography – PKC’2002*, volume 2274 of *LNCS*, pages 252 – 261. Springer-Verlag, 2002.
23. T. Okamoto and A. Shiraishi. A Digital Signature Scheme Based on Quadratic Inequalities. *Proceedings of Symposium on Security and Privacy*, pages 123 – 132, 1985.
24. V. Shoup. Number Theory C++ Library (NTL), version 5.0b. Available at <http://www.shoup.net>.
25. J. Stern, D. Pointcheval, J. Malone Lee, and P. Smart. Flaws in Applying Proof Methodologies to Signatures Schemes. In M. Yung, editor, *Advances in Cryptology – Crypto’02*, volume 2442 of *LNCS*, pages 93 – 110. Springer-Verlag, 2002.
26. B. Vallée, M. Girault, and P. Toffin. How to Break Okamoto’s Cryptosystem by Reducing Lattices Bases. In C. G. Günther, editor, *Advances in Cryptology – Eurocrypt’88*, volume 330 of *LNCS*, pages 281 – 291. Springer Verlag, 1988.

Efficient One-Time Proxy Signatures

Huaxiong Wang and Josef Pieprzyk

Centre for Advanced Computing – Algorithms and Cryptography
Department of Computing
Macquarie University
Sydney, NSW 2109, AUSTRALIA
{hwang,josef}@ics.mq.edu.au

Abstract. One-time proxy signatures are one-time signatures for which a primary signer can delegate his or her signing capability to a proxy signer. In this work we propose two one-time proxy signature schemes with different security properties. Unlike other existing one-time proxy signatures that are constructed from public key cryptography, our proposed schemes are based one-way functions without trapdoors and so they inherit the communication and computation efficiency from the traditional one-time signatures. Although from a verifier point of view, signatures generated by the proxy are indistinguishable from those created by the primary signer, a trusted authority can be equipped with an algorithm that allows the authority to settle disputes between the signers. In our constructions, we use a combination of one-time signatures, oblivious transfer protocols and certain combinatorial objects. We characterise these new combinatorial objects and present constructions for them.

1 Introduction

In general, digital signatures can be divided into two classes. The first class includes one-time signatures and their variants based on one-way functions without trapdoors. These schemes can be used to sign a predetermined number of messages only, we will call them *one/multiple-time signature schemes* (examples of such schemes includes one-time signatures by Lamport [16] and Rabin [27], but also multiple-time signatures by Rohatgi [32], by Reyzin and Reyzin [30], and by Pieprzyk, Wang and Xing [26]). The second class of schemes is based on public-key cryptography and they can be used to sign an unlimited number of messages. The RSA [29] and the ElGamal [10] signatures represent this class.

One-time signatures were first proposed by Rabin [27] and Lamport [16] and are based on the idea of committing public keys to secret keys using one-way functions. For more than 25 years, various variants of Rabin's schemes have been proposed and investigated by many researchers (see, for example, [3,4,11,16,20]). Indeed, one-time signatures have found many interesting applications [7,21], including on-line/off-line signatures [9], digital signatures with forward security properties [1], broadcast authentication protocols [25] and stream-oriented authentication [32] etc.

One of the main advantages of one-time signatures is their reliance on one-way functions without trapdoors that can be implemented using fast hash functions such as SHA-1 or MD5. The resulting signatures are the order of magnitude faster than signatures based on public cryptography. With the advent of low-powered, resource-constrained, small devices, such as cell phones, pagers, Palm pilots, smart cards etc. in recent years, one-time signatures have attracted more and more attention, as an attractive alternative to the traditional signatures based on public key cryptography (see, for example [15,25,30]).

Although digital signatures have been successfully applied to ensure the integrity, authenticity, and non-repudiation for the electronic documents, standard signatures (both based on public-key cryptography and on one-way functions) alone are too inflexible and inefficient to handle many practical requirements in new applications. Thus, many variants of the standard signatures with additional functionalities have been proposed. These include blind, undeniable, and group signatures to mention a few. Motivated by applications that require the power to sign to be transferred from one person to another, Mambo *et al* [19] proposed proxy signatures. Proxy signatures allow a designated person, called a *proxy*, to sign on behalf of a primary signer. A proxy signature convinces a verifier that the primary signer has delegated the signing power to the proxy and that the proxy has signed the message.

To our best knowledge, all the previously published proxy signatures are based on public-key cryptography. Most of the proxy signatures can be viewed as modifications of the ElGamal signature and their security typically relies on the assumption of the difficulty of the discrete logarithm problem (the DL assumption). In addition, these proxy schemes can generally be used for signing multiple messages and for multiple proxy signers.

In this paper, we will study *one-time* proxy signatures (or simply OTP signatures). As the name suggests, we consider one-time signatures with the additional *proxy* functionality. It should be noted that the notion of one-time proxy signature itself is not new, and it has been proposed by Kim *et al* [15] in a different context. Their signature is a variant of the ElGamal signature (or more precisely, a variant of one-time fail-stop signature [13]) and its security rests on the DL assumption. The motivation behind their work is to limit the power of the proxy signer so the proxy signer can sign once only. In contrast, our motivation is to enable the primary signer to delegate a proxy to sign in the applications where one-time signatures (based on one-way functions) are used.

To define our proxy signatures, we employ two basic cryptographic primitives as the building blocks. The first one is a one-time (or multiple-time) signature primitive based on one-way functions. The second building block is an oblivious transfer (OT) primitive. We then combine these primitives with certain combinatorial objects to obtain our OTP signatures. We formulate the general framework for proxy signatures, define their security goals and attacks against them. We then show that the efficiency of any OTP signature can be measured by the properties of the underlying combinatorial objects. We introduce *proxy patterns* that characterise the properties of these OTP signatures. Next, we give

constructions for the desired proxy patterns, using polynomials over finite fields and error-correcting codes, and link them with other combinatorial structures (such as Steiner systems).

The rest of the paper is organised as follows. In Section 2, we introduce our model of one-time proxy signatures. In Section 3, we consider candidates for the two building blocks that can be used to construct one-time proxy signatures. In Section 4, we propose a simple scheme for one-time proxy signatures and later we describe a basic scheme and analyse its security. In Section 5, we analyse the basic scheme and its security against the swallow attacks. Finally, Section 6 concludes the paper.

2 The Model

A *proxy signature* enables the primary signer to delegate his/her signing capability to a proxy signer so the proxy signer can generate a signature on behalf of the primary signer. Mambo *et al* [19] introduced the concept of proxy signature. They defined three classes of delegation: *full delegation*, *partial delegation* and *delegation by warrant*. A full delegation scheme assumes that the primary signer and the proxy signer have the same secret key, so the proxy signer can sign any message that is indistinguishable from the signature generated by the primary signer. A signature with partial delegation allows the primary signer to delegate the power of signing to a proxy in such a way that the signatures generated by the primary and proxy signers are different. This is normally done by making verification algorithms different for primary and proxy signatures. In other words, proxy signatures are distinguishable from primary signatures. A signature with delegation by warrant requires an additional piece of message (called a warrant) that determines the proxy signer that is delegated by the primary signer. Signatures with full delegation do not provide non-repudiation while signatures with partial delegation do. Signatures with delegation by warrant can be implemented using double signatures and therefore, they are not as efficient as signatures with full or partial delegations.

In this paper, we are interested in one-time signatures that allow full delegation with an added feature that allows to trace the authorship of the signature (if both proxy and primary signers agree to settle a dispute). Being more precise, we are going to consider proxy signatures with full delegation, in which the private signing key of the proxy signer is derived from the private key of the primary signer. In particular, we restrict our attention to signatures that can be used once only.

Informally, a *one-time proxy signature scheme* (OTP signature) includes two parties: a *primary signer* and a *proxy signer* together with the following three algorithms.

Key Generation: For a given security parameter, it outputs a pair of private and public keys for the primary signer and a private key for the proxy signer. The key generation may involve a two-party protocol run between the primary and proxy signers, or a multi-party protocol that is run amongst three parties: the primary signer, the proxy, and a trusted authority.

Singing: For an input that consists of a message to be signed and the private key of the signer (either primary or proxy), it outputs a valid signature.

Verifying: For an input that includes a pair (a message and a signature) and the public key of the primary signer, it outputs either *accept* or *reject*.

In the following, we consider the basic security requirements imposed on OTP signatures. If an OTP signature satisfies the requirements, it is called *secure*.

Unforgeability: It is infeasible for any third party (that has not been involved in signing) to forge a message/signature that passes the signature verification. This means that if a signature has been generated by the primary signer, no body (including the proxy) can forge a message/signature. Also if the signature has been generated by the proxy, then no body (including the primary signer) can forge a message/signature.

Verifiability: For a valid signature, a verifier is convinced that the primary signer has agreed to sign a message (either the primary signer has signed it or the proxy has).

Traceability: In case of a dispute between the primary and proxy signers, there exists a tracing algorithm that reveals the identity of the actual signer. That is, the algorithm guarantees that it should be infeasible for

- the primary signer to sign a message and to claim later that it has been signed by the proxy signer.
- the proxy signer to sign a message and to claim later that it has been signed by the primary signer.

We note that the model of our OTP signature is slightly different from previous proxy signatures in the sense that there is only one public key of the primary signer for the signature verification. Thus, from a verifier point of view, signatures generated by primary or proxy signers are indistinguishable (like in the full delegation). However, the tracing algorithm guarantees the non-repudiation property for the primary signer and the proxy signer. Thus, unlike in full delegation signatures, the primary signer and the proxy signer have different private keys for signature generation, and in case a dispute occurs between the two potential signers, the tracing algorithm is called to resolve it. We argue that the indistinguishable between the signatures by the primary signer and the proxy signer is an interesting property, for example, it can be used to protect the privacy of the actual signer. However, in this paper we are not going to explore it beyond this point.

3 Building Blocks

In this section, we review two cryptographic primitives that are needed in the our constructions of proxy signatures.

3.1 One-Time Signature

One-time signatures are based on one-way functions. Rabin published the first one-time signature based on a private-key encryption or a one-way function

without a trapdoor [27], requiring interaction between the signer and the verifier. Lamport [16] gave a non-interactive one-time signature using a one-way function. The idea of Lamport is as follows. For a given one-way function f , one selects two random strings x_0, x_1 as the secret key, and publishes $f(x_0)$ and $f(x_1)$ as the public key. Then the single-bit message $b \in \{0, 1\}$ can be signed by revealing x_b . Various modifications of the Lamport signature with improved efficiency and functionalities have been proposed (see, for example [2,4,5,9,12,14,21,25,30,32]).

As our building block, we are going to use a one-time signature defined as follows. Let b, t, k be integers such that $\binom{t}{k} \geq 2^b$. Let T denote the set $\{1, 2, \dots, t\}$ and \mathcal{T}_k be the family of k -subsets of T . Let S be a one-to-one mapping from $\{0, 1, \dots, 2^b - 1\}$ to \mathcal{T}_k such that for a message m , $S(m)$ assigns a unique k -element subset from \mathcal{T}_k . Let f be a one-way function operating on ℓ -bit strings (ℓ is a security parameter).

The signature scheme consists of three algorithms: *key generation*, *signing* and *verification*. For a given security parameter ℓ , the key generator chooses at random t strings s_i of the length ℓ bits and creates the secret key $SK = (s_1, \dots, s_t)$. The public key is the image of the secret key obtained using the one-way function f , i.e., $PK = (v_1, \dots, v_t)$ such that $v_1 = f(s_1), \dots, v_t = f(s_t)$.

To sign a b -bit message m , the signer interprets m as an integer between 0 and $2^b - 1$ and computes $S(m) = \{i_1, \dots, i_k\} \in \mathcal{T}_k$. The value s_{i_1}, \dots, s_{i_k} is the signature of m .

To verify a signature $(s'_1, s'_2, \dots, s'_k)$ on a message m , the verifier again interprets m as an integer between 0 and $2^b - 1$ and computes $\{i_1, \dots, i_k\}$ as the m -th k -element subset of \mathcal{T}_k . Finally, the verifier checks whether $f(s'_1) = v_{i_1}, \dots, f(s'_k) = v_{i_k}$.

Definition 1. We call the above one-time signature scheme a (t, k) one-time signature scheme and denote it by $\mathcal{O} = (T, S, f)$, or simply by \mathcal{O} . The parameters (t, k) specify efficiency of the signature.

Note that the Bos-Chaum one-time signature scheme [2] is a special case of the (t, k) scheme in which $k = t/2$. Note also that for a (t, k) one-time signature $\mathcal{O} = (T, S, f)$, the most expensive part of computation is the implementation of the mapping S . In [30], Reyzin and Reyzin present two algorithms for implementation for S with computation costs of $O(tk \log^2 t)$ or $O(k^2 \log t \log k)$. In [26], Pieprzyk *et al* give more efficient implementations for S through the explicit constructions of S using polynomials over finite fields, error-correcting codes, and algebraic curves.

3.2 Oblivious Transfer (OT)

An oblivious transfer (OT) refers to a two-party protocol executed between a sender S and a receiver R . The goal of the protocol is to transfer the knowledge about an input string held by the sender to the receiver in such a way that the receiver learns some part of the input but the sender cannot figure out which part of the input is now known to the receiver. Consider a 1-out- n oblivious

transfer (OT_1^n) protocol. The sender S has n secrets (strings) m_1, m_2, \dots, m_n , and is willing to disclose one of them (m_α) to R for some index α chosen by R . However, R does not want to reveal its choice of the index α to S and at the same time, S does not want R to gain any information about other secrets $m_i, i \neq \alpha$. In general, we may have a k -out- n oblivious transfer (OT_k^n), in which R may choose k indices out of n .

The concept of oblivious transfer has been introduced by Rabin in 1981 [28] and it has been extensively studied (see, for example, [8,22,23]). Here is an example of OT_1^n proposed recently by Tzeng [33], which is among the most efficient OT protocols proposed so far. Let g and h be two (public) generators in a q -order group G_q , where q is prime. Assume that the secret input of S is $m_1, m_2, \dots, m_n \in G_q$, and the choice of R is α , $1 \leq \alpha \leq n$. The protocol proceeds as follows.

1. $R \rightarrow S : y = g^r h^\alpha$ for a random $r \in \mathbb{Z}_q$,
2. S randomly chooses n elements $k_i \in \mathbb{Z}_q$ and

$$S \rightarrow R : c_i = (g^{k_i}, m_i(y/h^i)^{k_i}), \quad 1 \leq i \leq n.$$

3. R computes $m_\alpha = b/a^r$, assuming $c_\alpha = (a, b)$.

It is proved in [33] that in the above OT_1^n protocol, the confidentiality of the receiver choice is unconditionally secure and the confidentiality of un-chosen secrets is at least as strong as the hardness of the decision Diffie-Hellman problem. As to computations required in the protocol, the receiver needs to compute 2 modular exponentiations and the sender computes $2n$ modular exponentiations.

4 One-Time Proxy Signatures

Our basic idea behind the constructions of OTP signatures is as follows. The primary signer generates n private/public key pairs for one time signatures, say $(sk_1, pk_1), \dots, (sk_n, pk_n)$. The proxy signer gains one of the n private keys, say sk_i in such a way that the primary signer does not know, which key was obtained by the proxy signer, i.e., the primary signer does not know the index i . The primary signer publishes the public key pk_1, \dots, pk_n in an authenticated way. The proxy signer uses sk_i to sign the message, which can be verified by anyone who knows the public key. Note that the verification of signatures generated by primary and proxy signers is the same.

To prevent cheating by signers, a tracing algorithm has to be carefully designed. The algorithm should be run by a trusted authority and should identify the true signer with a high probability. Note that the oblivious transfer enables us to identify the true signer. To do this, the trusted authority always asks the proxy to sign the disputed message again. If the proxy is unable to produce a different signature it means that either the proxy really signed the message or the primary signer has applied the same secret key as proxy (this event happens with the probability $1/n$).

4.1 A Simple Proxy Signature Scheme

We present a simple and somewhat trivial scheme to illustrate the basic idea. Then we improve its efficiency using some combinatorial techniques. The scheme is based on a (t, k) one-time signature $\mathcal{O} = (T, S, f)$ and an oblivious transfer protocol OT_1^n (or OT_k^n), and it works as follows.

Key Generation: It consists of the following three steps.

- The primary signer randomly chooses an $n \times t$ array $A = (s_{ij})_{n \times t}$ as her private key. Each row holds t secret keys of an instance of the (t, k) one-time signature \mathcal{O} . The public key is $V = (v_{ij})_{n \times t}$, where $v_{ij} = f(s_{ij})$ and f is the one-way function from \mathcal{O} .
- The primary and proxy signers execute an OT_1^n (or OT_k^n) protocol. At the end of the protocol, the proxy signer learns one row from A , say (s_{i1}, \dots, s_{it}) , as his private key, but nothing more. The primary signer has no information about the index i .
- The proxy signer applies f to (s_{i1}, \dots, s_{it}) and compares the results with the i th row of public array V . If the check fails to hold, the proxy exits the scheme and complains to the primary signer.

Signing: The proxy signer applies the i th row of A , i.e., (s_{i1}, \dots, s_{it}) , as his private key of the one-time signature \mathcal{O} and signs the message m . That is the proxy signer first computes $S(m) = \{j_1, \dots, j_k\} \subseteq \{1, \dots, t\}$ and then reveals m and the signature $\delta = \{(s_{ij_1}, \dots, s_{ij_k}), i\}$.

Verifying: This part follows the steps necessary to verify an instance of the (t, k) one-time signature.

Security. We discuss the security requirements of the scheme. Obviously, unforgeability and verifiability of the OTP signature follow directly from the unforgeability and verifiability of the underlying one-time signature \mathcal{O} . What we need to consider is the traceability of the true signer (in case of cheating attempts from either the proxy or the primary signer).

Unforgeability against the primary signer: Assume that the primary signer wants to cheat. She generates a signature for a message m and later claims that it was generated by the proxy signer. Note that to sign m , the primary signer has to choose a row of A and to sign using the chosen instance of one-time signature. Suppose that she has chosen j th row of A . The generated signature is $\delta_j = \{(s_{ji_1}, \dots, s_{ji_k}), j\}$, where $S(m) = \{i_1, \dots, i_k\}$. The proxy signer can prove that the signature was not generated by him, by revealing another signature for m using his private key (s_{i1}, \dots, s_{it}) . That is, he reveals the signature $\delta_i = \{(s_{ii_1}, \dots, s_{ii_k}), i\}$, which shows that $\delta_i \neq \delta_j$. As the proxy signer knows only one row of the private keys, he can only sign the message with one of the rows, so δ_j must have been generated by the primary signer. The OT protocol provides unconditional security for the proxy signer and the probability of success of the primary signer is $1/n$.

Unforgeability against the proxy signer: Suppose that the proxy signer wants to cheat, he generates a signature, later denies it and claims that the primary signer

(or someone else) has generated the signature. His claim can be accepted only if he can generate a different signature for the same message. In other words, the proxy is able to produce two different signatures for the same message. This is impossible unless, he is able to break the OT protocol or to invert the one-way function.

We stress that the tracing algorithm is called only if the dispute between the primary signer and the proxy signer occurs. The knowledge of a valid signature alone is not sufficient to identify the actual signer (the signature provides full delegation).

Efficiency. We look at the efficiency of the scheme. The signing and verification of the signature are exactly the same as the underlying one-time signature scheme, so could be very fast. The key generation requires n times costs of key generation for one-time signatures, plus the cost of running an OT_1^n (or OT_t^n) protocol. The length of public and secret keys increases n times as well. However, observe that the key generation, which is the most expensive part of computations, can be precomputed. Furthermore, an expensive OT protocol can be avoided if a third trusted party helps during the key generation. The private key of the primary signer can be discarded after the key generation. In the next section we propose methods to reduce the public key length.

4.2 The Basic Proxy Signature Scheme

To decrease the probability of successful cheating by the primary signer, it is required to increase the parameter n and consequently the number of rows in A . This causes that the simple proxy signature secure against a dishonest primary signer must have a long private/public key. We show that the simple proxy signatures can be converted into proxy signatures with shorter public keys using combinatorial techniques.

Definition 2. Given a set $X = \{x_1, \dots, x_M\}$ and an $n \times t$ array $C = [c_{ij}]$ with entries from X . The array C is called a (t, k, n, M) proxy pattern, denoted by $PP(t, k, n, M)$, for a (t, k) one-time signature if

1. each row of C contains t different elements of X ,
2. any two distinct rows of C have at most $k - 1$ common elements, i.e., for any $i \neq j$,

$$|\{c_{i1}, \dots, c_{it}\} \cap \{c_{j1}, \dots, c_{jt}\}| < k.$$

For a given $PP(t, k, n, M)$, we combine it with a (t, k) one-time signature to construct an OTP signature that is a generalisation of the simple scheme presented above. Without the loss of generality, assume that $C = (c_{ij})$ is a $PP(t, k, n, M)$ with entries taken from $X = \{1, \dots, M\}$ and $\mathcal{O} = (T, S, f)$ is a (t, k) one-time signature. Our basic proxy signature works as follows.

Key Generation: It goes through the following three steps.

- The primary signer randomly chooses M distinct values (s_1, s_2, \dots, s_M) as the private key (for example, each s_i is an ℓ -bit string if the underlying one-time signature \mathcal{O} is defined for the security parameter ℓ). The public key is $V = (v_1, \dots, v_M)$, where $v_i = f(s_i), i = 1, \dots, M$.

- The primary and proxy signers execute an OT_t^M protocol. At the end of the protocol, the proxy signer learns the i th row of C , that is $(s_{c_{i1}}, \dots, s_{c_{it}})$, as his private key, but nothing more. The primary signer has no information about the index i .
- The proxy signer applies f to $(s_{c_{i1}}, \dots, s_{c_{it}})$ and checks the results with the corresponding components of the public key V . If the check fails, the proxy aborts and complains.

Signing: For a given message m , the proxy signer applies his private key $(s_{c_{i1}}, \dots, s_{c_{it}})$ to the one-time signature \mathcal{O} and signs the message. That is, the proxy signer first computes $S(m) = \{j_1, \dots, j_k\} \subseteq \{1, \dots, t\}$ and then reveals the signature $\delta = \{(s_{c_{ij_1}}, \dots, s_{c_{ij_k}}), i\}$.

Verifying: It follows the verification of the (t, k) one-time signature (applied to the appropriate instance of the one-time signature) in a straightforward manner.

It is easy to see that the security of this scheme is similar to the security of the simple scheme. The traceability is guaranteed by the properties of the proxy pattern C , that is, any two rows will have at most $k-1$ common elements. Since a signature requires the knowledge of k secret values of the private key, the proxy signer can resolve disputes by showing two valid signatures (corresponding to two different rows of C).

The main advantage of the basic signature scheme is a reduction of the length of public key (and the corresponding private key) from nt values to M values. In the remainder of this section, we will give constructions for proxy patterns with small M and derive a bound on the minimal value for M .

4.3 Constructions of Proxy Patterns

It is easy to see that the simple signature uses a trivial $PP(t, k, n, nt)$ for any $k, 1 \leq k \leq t$. By fixing k , as this is the case for the underlying (k, t) one-time signature, we are able to construct a $PP(t, k, n, M)$ such that M is significantly smaller than nt , and so to reduce the length of the public key.

Assume $GF(q)$ is a finite field with q elements and a_1, \dots, a_t are t distinct elements from $GF(q)$. We construct a $PP(t, k, n, M)$ as follows. Consider a set $X = \{a_1, \dots, a_t\} \times GF(q)$ and all polynomials of the degree at most $k-1$ over $GF(q)$. Next write them as $g_1(x), \dots, g_{q^k}(x)$. Note that there are q^k such polynomials. Further define a $q^k \times t$ array $C = [c_{ij}]$ with entries taken from X , so

$$c_{ij} = (a_j, g_i(a_j)), \quad \text{for } i = 1, 2, \dots, q^k, j = 1, 2, \dots, t.$$

Now we show that C is a $PP(t, k, q^k, qt)$. Indeed, for $1 \leq i \leq q^k$, the i th row of C is

$$((a_1, g_i(a_1)), (a_2, g_i(a_2)), \dots, (a_t, g_i(a_t))).$$

Thus, for $i \neq j$,

$$\begin{aligned}
 & |\{(a_1, g_i(a_1)), \dots, (a_t, g_i(a_t))\} \cap \{(a_1, g_j(a_1)), \dots, (a_t, g_j(a_t))\}| \\
 &= |\{a \mid g_i(a) = g_j(a)\}| \\
 &= |\{a \mid (g_i - g_j)(a) = 0\}| \\
 &< k
 \end{aligned}$$

otherwise there are k or more than k roots for the polynomial $g_i - g_j$. But $g_i - g_j$ is a polynomial of degree at most k , it follows that $g_i = g_j$ which contradicts that $i \neq j$. We have proved the following result.

Theorem 1. *Let q be a prime power. For any integers t, k such that $k \leq t \leq q$, there exists a $PP(t, k, q^k, qt)$.*

Note that for the simple proxy signature, a $PP(t, k, q^k, q^{kt})$ is required. Thus, for the fixed parameters t, k and q^{k+1} , we can reduce the number of elements in the public key from q^{kt} for the simple proxy signature to qt in the basic proxy signature.

A generalisation of the above polynomial construction uses error-correcting codes. Let Y be an alphabet of q elements. An (N, W, D, q) code is a set \mathcal{M} of W vectors in Y^N such that the Hamming distance between any two distinct vectors in \mathcal{M} is at least D . Consider an (N, W, D, q) code \mathcal{M} . We write each codeword as $m_i = (m_{i1}, \dots, m_{iN})$ with $m_{ij} \in Y$, where $1 \leq i \leq W, 1 \leq j \leq N$. For a set $X = \{1, \dots, N\} \times Y$, we define a proxy pattern $C = (c_{ij})$ as follows,

$$c_{ij} = (j, m_{ij}), \quad \text{for } i = 1, 2, \dots, W, j = 1, 2, \dots, N.$$

Now for each distinct i, j , we have

$$\begin{aligned}
 & |\{c_{i1}, c_{i2}, \dots, c_{iN}\} \cap \{c_{j1}, c_{j2}, \dots, c_{jN}\}| \\
 &= |\{(k, m_{ik}) : 1 \leq k \leq N\} \cap \{(k, m_{jk}) : 1 \leq k \leq N\}| \\
 &= |\{k : m_{ik} = m_{jk}\}| \\
 &< N - D + 1.
 \end{aligned}$$

This shows that the array C constructed above is a $PP(N, N - D + 1, W, Nq)$. We then have

Theorem 2. *If there exists an (N, W, D, q) code, then there exists a $PP(N, N - D + 1, W, Nq)$.*

In the coding theory, it is known that for given k and q there are constructions (e.g. using algebraic geometry codes [24]) for (N, W, D, q) codes for which $N = O(\log W)$. In the context of proxy patterns, this means that there exists $PP(N, N - D, W, Nq)$ in which $N = O(\log W)$. Applying this observation to one-time proxy signature, we can reduce the number of elements in the public key from $O(n)$, for the simple proxy signature, to $O(\log n)$ for the proxy signature based on the coding construction.

4.4 Bounds for Proxy Patterns

To minimise the success probability of cheating by the primary signer, we need to have a $PP(t, k, n, M)$ for which the value n is as large as possible while other parameters t, k and M are fixed. In the following we derive an upper bound for such n .

Theorem 3. *For any $PP(t, k, n, M)$, the following inequality holds*

$$n \leq \frac{\binom{M}{k}}{\binom{t}{k}}.$$

Proof. Assume that $C = [c_{ij}]$ is a $PP(t, k, n, M)$ with entries taken from an M -set of X . For each row i , we associate a subset B_i of X , i.e., $B_i = \{c_{i1}, \dots, c_{it}\} \subseteq X$, where $i = 1, \dots, n$. Clearly, $|B_i| = t$ and $|B_i \cap B_j| < k$ for all i, j where $i \neq j$. For each $1 \leq i \leq n$, denote \mathcal{R}_i to be the family of all the k -subsets of B_i . This implies that $|\mathcal{R}_i| = \binom{t}{k}$. Now we claim that $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$ for each $i \neq j$. If this claim is not true or $B \in \mathcal{R}_i \cap \mathcal{R}_j$ is a k -subset of X , then B is a k -subset of both B_i and B_j , which contradicts the fact that $|B_i \cap B_j| < k$. Thus we have

$$\binom{M}{k} \geq |\cup_{i=1}^n \mathcal{R}_i| = n|\mathcal{R}_i| = n\binom{t}{k}.$$

The desired result follows immediately. \square

Next, we show that the bound in Theorem 3 is tight for some parameter set. Recall that a *Steiner system* $S(k, t, M)$ is a pair (X, \mathcal{B}) , where X is a set of M elements called *points* and \mathcal{B} is a family of t -subsets of X called *blocks*, such that every k -subset of points is contained in a unique block. It is known that the number of blocks of an $S(k, t, M)$ is $\binom{M}{k}/\binom{t}{k}$.

Corollary 1. *An $PP(t, k, n, M)$ with $n = \binom{M}{k}/\binom{t}{k}$ exists if and only if there exists an $S(k, t, M)$.*

Proof. Let (X, \mathcal{B}) be an $S(k, t, M)$. For each block, associate a row of an $n \times t$ array in a natural way, i.e., entries of the i th row are assigned to the elements in the block B_i . It is easy to see that assignment gives rise to a $PP(t, k, n, M)$ with $n = \binom{M}{k}/\binom{t}{k}$.

On the other hand, assume that C is a $PP(t, k, \binom{M}{k}/\binom{t}{k}, M)$ with entries from M -set X , each row of C is a subset of X , we obtain a set system (X, \mathcal{B}) where $\mathcal{B} = \{B_i : 1 \leq i \leq \binom{M}{k}/\binom{t}{k}\}$. It is clear that each k -subset of X appears in at most one block. So we need to show that it is contained in at least one block. Using the same notation as in Theorem 3, we know that each block B_i contributes $\binom{t}{k}$ k -subsets \mathcal{R}_i of X . Since \mathcal{R}_i s are disjoint and there are $\binom{M}{k}/\binom{t}{k}$ such \mathcal{R}_i , which gives rise all the $\binom{M}{k}$ possible choices of k -subsets of X , that means that any k -subset must be in one of the \mathcal{R}_i . This concludes the proof. \square

5 Proxy Signatures Secure against Swallow Attacks

Consider the following attack: suppose the primary signer has seen a valid signature (m, δ) produced by the proxy. She knows that the private key of the proxy signer is the i th row of the proxy pattern. Now the primary signer *swallows* the signature generated by the proxy signer, and generates the signature for another new message, using the private key of the proxy signer. In this case, the proxy signer is unable to prove his innocence. We will call it, the *swallow attack*.

In order to protect proxy signatures against the swallow attack, the primary signer should not be able to guess the private key of the proxy from a signature produced by the proxy. Looking at a message and its signature, the primary signer should not be able to determine the private key of the proxy. In other words, a single proxy signature should point at many (potential) private keys of the proxy. On the other hand, there should not be *too many* private keys corresponding to a given proxy signature. Otherwise, the proxy signature can be subject to an attack in which the primary signer chooses at random the proxy private key (without looking at the signature) and succeeds with a high probability. Based on this observation, we propose a new proxy signature that is secure against the swallow attack.

First we need some notation. Let $C = (c_{ij})$ be an $n \times t$ array with entries from an M -set of X . For any $1 \leq i \leq n$ and $1 \leq j_1 \leq j_2 \leq \dots \leq j_k \leq t$, we denote

$$C[i; j_1, j_2, \dots, j_k] = \{\ell \mid c_{\ell j_1} = c_{ij_1}, \dots, c_{\ell j_k} = c_{ij_k}\}.$$

In other words, $C[i; j_1, j_2, \dots, j_k]$ is the set of indices of the rows which are identical to i th row when restricted to the j_1, \dots, j_k columns.

Definition 3. Given a set $X = \{x_1, \dots, x_M\}$. An $n \times t$ array $C = (c_{ij})$, with entries from X , is called a (λ_1, λ_2) -strong (t, k, n, M) proxy pattern, denoted by (λ_1, λ_2) -SPP (t, k, n, M) for a (t, k) one-time signature if

1. each row of C contains t different elements of X ,
2. any two distinct rows of C have at most k common elements, i.e., for any $i \neq j$,

$$|\{c_{i1}, \dots, c_{it}\} \cap \{c_{j1}, \dots, c_{jt}\}| \leq k.$$

3. for any row $1 \leq i \leq n$ and any k columns $1 \leq j_1 \leq j_2 \leq \dots \leq j_k \leq t$,

$$\lambda_1 \leq |C[i; j_1, j_2, \dots, j_k]| \leq \lambda_2.$$

We now combine a (λ_1, λ_2) -SPP (t, k, n, M) and a (t, k) one-time signature to construct an OTP signature secure against the swallow attack. Assume $C = (c_{ij})$ is a (λ_1, λ_2) -SPP (t, k, n, M) with entries taken from $X = \{1, \dots, M\}$ and $\mathcal{O} = (T, S, f)$ is a (t, k) one-time signature. The signature works as follows.

Key Generation: It consists of the following three steps.

- The primary signer randomly chooses M distinct elements (s_1, s_2, \dots, s_M) as the private key (for example, each s_i is a ℓ -bit string if the private key of underlying one-time signature \mathcal{O} consists of ℓ -bits strings). The public key is $V = (v_1, \dots, v_M)$, where $v_i = f(s_i), i = 1, \dots, M$.

- The primary and proxy signers execute an OT_t^M protocol. At the end of the protocol, the proxy signer learns a t -subset of X that is the i th row of C , i.e., $(s_{ci1}, \dots, s_{cit})$, as his private key, but nothing more. The primary signer has no information about the index i .
- The proxy signer applies f to $(s_{ci1}, \dots, s_{cit})$ and checks the results by comparing them to the corresponding components of the public key V . If the check fails, the proxy aborts and complains.

Signing: To sign a message m , the proxy signer computes $S(m) = \{j_1, j_2, \dots, j_k\}$ and $C[i; j_1, \dots, j_k]$. Then he randomly chooses $\ell \in C[i; j_1, \dots, j_k]$, and reveals $\delta = \{(s_{c_{\ell j_1}}, \dots, s_{c_{\ell j_k}}), \ell\}$ as the signature.

Verifying: It follows the verification of the (t, k) one-time signature (applying to the ℓ th row) in a straightforward manner.

Clearly, the unforgeability against the third party is the same as the underlying one-time signature scheme \mathcal{O} . Next we show that the scheme is secure against regular attacks and the swallow attacks from the primary signer.

Lemma 1. *The probability that the primary signer succeeds in the regular attack (without seeing any signature) is at most λ_2/n .*

Proof. In this attack, the primary signer generates a signature and later claims that it is generated by the proxy signer. She succeeds if the proxy signer fails to prove that he has not generated the signature. As the primary signer has no information about the index i chosen by the proxy signer, she may try to guess it. Assume that she has chosen the index j . For a message m , the primary signer computes $S(m) = \{j_1, \dots, j_k\}$ and reveals the signature $\{s_{c_{jj_1}}, \dots, s_{c_{jj_k}}, \ell\}$, where $\ell \in C[j; j_1, \dots, j_k]$. Note that if $j \notin C[i; j_1, \dots, j_k]$, then the proxy can sign the message m using a different key from the i th row, which results in different signature of the primary signer. The primary signer succeeds if and only if $j \in C[i; j_1, j_2, \dots, j_k]$. Since C is a (λ_1, λ_2) -SPP($t, k + 1, n, M$), we know that $|C[i; j_1, \dots, j_k]| \leq \lambda_2$ and the result follows. \square

Lemma 2. *The probability that the primary signer succeeds in the swallow attack (having seen a signature) is at most $\max\{1/\lambda_1, \lambda_2/n\}$.*

Proof. In this attack, the primary signer has seen a message/signature pair (m, δ) generated by the proxy signer. Next she swallows the data and generates another message/signature pair (m', δ') . She succeeds if the proxy signer fails to prove that there is a cheating from the primary signer. Suppose that the proxy signer has chosen the index i . For a signature (m, δ) generated by the proxy signer, we may assume that $\delta = \{(s_{c_{\ell j_1}}, \dots, s_{c_{\ell j_k}}), \ell\}$, where $S(m) = \{j_1, \dots, j_k\}$ and $\ell \in C[i; j_1, \dots, j_k]$. Having seen the signature δ , the primary signer knows that the secret index chosen by the proxy signer is one of the elements in $C[\ell; j_1, \dots, j_k]$. One attack strategy from the primary signer is to randomly choose $j \in C[\ell; j_1, \dots, j_k]$ and use secret key from j th row to generate the signature (m', δ') . She succeeds with probability $1/|C[\ell; j_1, \dots, j_k]|$ that $j = i$. If $j \neq i$, then the proxy signer can generate the signature for m' , say δ'' . It can

be seen that $\delta' \neq \delta''$, which means that the proxy can create two signatures for the same message m' using two different row keys. This proves that the primary signer attempted to cheat. Another strategy for the primary signer is to choose $j \notin C[\ell; j_1, \dots, j_k]$. In this case, she succeeds if and only if $j \in C[i; j'_1, \dots, j'_k]$, where $S(m') = \{j'_1, \dots, j'_k\}$. As in the proof of Lemma 1, the probability of a successful attack using this strategy is at most λ_2/n . Therefore, the overall success probability of the attack is bounded by $\max\{1/\lambda_1, \lambda_2/n\}$. \square

Previously, we have used polynomials over a finite field to construct a $PP(t, k, q^k, qt)$. We will show that this construction can be extended for (q, q) -SPP($t, k-1, q^k, qt$).

Theorem 4. *The polynomial construction for a $PP(t, k, q^k, qt)$ given in Section 4 results in a (q, q) -SPP($t, k-1, q^k, qt$).*

Proof. We already know that the polynomial construction gives rise to a $PP(t, k, q^k, qt)$, $C = (c_{ij})$. To show that C is a (q, q) -SPP($t, k-1, q^k, qt$). We need to show that for any $1 \leq i \leq q^k$ and $1 \leq j_1 \leq j_2 \dots, j_{k-1} \leq t$, we have

$$C[i; j_1, j_2, \dots, j_{k-1}] = q.$$

In other words, we need to show that for any $k-1$ distinct elements $a_{j_1}, \dots, a_{j_{k-1}} \in GF(q)$, and any $k-1$ elements $\alpha_1, \dots, \alpha_{k-1} \in GF(q)$, there are exactly q polynomials g of degree at most $k-1$ such that

$$g(a_{i_1}) = \alpha_1, \dots, g(a_{i_{k-1}}) = \alpha_{k-1}. \quad (1)$$

Indeed, choose $a \in GF(q) \setminus \{a_{i_1}, \dots, a_{i_{k-1}}\}$, then a polynomial g satisfying (1) is uniquely determined by the value of $g(a)$, there are q different possible choices for $g(a)$ which in turn give rise to q possible polynomial polynomials satisfying (1). This proves our desired result. \square

It should be noted that constructions for strong proxy patterns can also be based on error-correcting codes. The argument follows the one developed in Section 4.3. However, it is not clear how the parameters λ_1, λ_2 are related to the parameters of the codes. We believe that it is an interesting problem for further research.

6 Conclusions

In this work, we have studied one-time proxy signature schemes. Unlike other existing one-time proxy signature scheme that are constructed using public-key cryptography, we have proposed one-time proxy signatures based on one-way functions. These signatures preserve the basic functionalities and properties of one-time signatures (including their fast generation and verification) but also allow the primary signer to delegate the power of signing to a chosen proxy.

The one-time proxy signatures permit full delegation for which potential verifiers are not able to distinguish primary signers from proxy. However, in case

of a dispute between the signers about the authorship of a signature, a trusted authority is able to run an algorithm to resolve the dispute. The algorithm asks the proxy to re-generate a signature for the disputed message. If the proxy is able to produce a signature different from the disputed one, then the true signer of the signature is the primary signer. Otherwise, the proxy has generated the signature.

One-time proxy signatures can be especially useful where there is a need for fast generation and verification together with a need to share power of signing. Applications may include authentication of streams of packets in a distributed environment with mirror servers generating proxy signatures.

Our approach is based on a combination of certain type of existing one-time signature with some combinatorial objects. While the former can be optimised using the known techniques in the literature, the latter are new combinatorial objects we introduce in this paper and so are of independent interest. In particular, the structures of strong proxy patterns are far from clear, and providing efficient constructions for them is an interesting research problem.

Acknowledgement

The work was in part supported by Australian Research Council Discovery grants DP0345366 and DP0344444.

References

1. M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme, *Advances in Cryptology – Asiacrypt’00*, LNCS, **1976**(2000), 116-129.
2. J. N. E. Bos and D. Chaum. Provably unforgeable signature, *Advances in Cryptology – Crypto’92*, LNCS, **740**(1993), 1-14.
3. M. Bellare and S. Micali. How to sign given any trapdoor function. *Journal of Cryptology*, **39**(1992), 214-233.
4. D. Bleichenbacher and U. Maurer. Directed acyclic graphs, one-way functions and digital signatures, *Advances in Cryptology – Crypto’94*, LNCS, **839**(1994), 75-82.
5. D. Bleichenbacher and U. Maurer. On the efficiency of one-time digital signatures, *Advances in Cryptology – Asiacrypt’96*, LNCS, **1163**(1996), 145-158.
6. D. Bleichenbacher and U. Maurer. Optimal tree-based one-time digital signature schemes, *STACS’96*, LNCS, **1046**(1996), 363-374.
7. C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications, *Advances in Cryptology – Crypto’94*, LNCS, **839**(1994), 234-246.
8. G. Di Crescenzo, T. Malkin and R. Ostrovsky. Single database private information retrieval implies oblivious transfer, *Advances in Cryptology – Eurocrypt’00*, LNCS, 2000, 122-138.
9. S. Even, O. Goldreich and S. Micali. On-line/off-line digital signatures, *Journal of Cryptology*, **9**(1996), 35-67.
10. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory*. **31**(1985), 469-472.
11. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, **17**(1988), 281-308.

12. A. Hevia and D. Micciancio. The provable security of graph-based one-time signatures and extensions to algebraic signature schemes. *Advances in Cryptology – Asiacrypt’02*, LNCS, **2501**(2002), 379-396.
13. T. P. Pedersen and B. Pfitzmann. Fail-stop signatures. *SIAM Journal on Computing*, **26/2**(1997), 291–330.
14. Y.-C Hu, A. Perrig and D.B. Johnson. Packet Leashes: A defense against wormhole attacks in wireless Ad Hoc Networks. *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, 2003, to appear.
15. H. Kim, J. Baek, B. Lee and K. Kim. Secret Computation with secrets for mobile agent using one-time proxy signature. The 2001 Symposium on Cryptography and Information Security, Oiso, Japan.
16. L. Lamport. Constructing digital signatures from a one way function. *Technical Report CSL-98*, SRI International, 1979.
17. L. Lamport. Password authentication with insecure communication. *Communication of the ACM*, **24**(11), 1981, 770-772.
18. B. Lee, H. Kim and K. Kim. Strong proxy signature and its applications. The 2001 Symposium on Cryptography and Information Security, Oiso, Japan.
19. M. Mambo, K. Usuda and E. Okamoto. Proxy signatures: Delegation of the power to sign messages. *IEICE Trans. Fundamentals*, Vol. E79-A (1996), 1338-1353.
20. R.C. Merkle. A digital signature based on a conventional function. *Advances in Cryptology – Crypto’87*, LNCS, **293**(1987), 369-378.
21. R.C. Merkle. A certified digital signature. *Advances in Cryptology – Crypto’87*, LNCS, **435**(1990), 218-238.
22. M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. *Proceedings of the 31st ACM Symposium on Theory of Computing*, 1999, 245-254
23. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. SODA01, 2001.
24. H. Niederreiter and C. P. Xing, *Rational Points on Curves over Finite Fields: Theory and Applications*, Cambridge University Press, LMS 285, 2001.
25. A. Perrig. The BiBa one-time signature and broadcast authentication. *Eighth ACM Conference on Computer and Communication Security*, ACM, 2001, 28-37.
26. J. Pieprzyk, H. Wang and C. Xing. Multiple-time signature schemes secure against adaptive chosen message attacks. *the 10th annual workshop on Selected Areas in Cryptography (SAC03)*, LNCS, to appear.
27. M.O. Rabin. Digitalized signatures. *Foundations of Secure Communication*, Academic Press, 1978, 155-168.
28. M.O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard University, 1981.
29. R.L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, **21**(1978), 120-12.
30. L. Reyzin and N. Reyzin. Better than BiBa: Short one -time signatures with fast signing and verifying. *Information Security and Privacy (ACISP02)*, LNCS, **2384**(2002), 144-153.
31. R. Rivest and A. Shamir. PayWord and MicroMint: two simple micro payment schemes. *Tech. Rep.*, MIT Lab. for Computer Science, 1996.
32. P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. *6th ACM conference on Computer and Communication Security*, 1999, 93-100.
33. W-G Tzeng. Efficient 1-out- n Oblivious Transfer Schemes. PKC’02, LNCS, 159-171.

Universal Designated-Verifier Signatures

Ron Steinfeld¹, Laurence Bull², Huaxiong Wang¹, and Josef Pieprzyk¹

¹ Dept. of Computing, Macquarie University, North Ryde, NSW 2109, Australia
{rons, hwang, josef}@comp.mq.edu.au
<http://www.ics.mq.edu.au/acac/>

² School of Computer Science and Software Engineering
Monash University, Melbourne, Australia
{laurence.bull@csse.monash.edu.au}

Abstract. Motivated by privacy issues associated with dissemination of signed digital certificates, we define a new type of signature scheme called a ‘Universal Designated-Verifier Signature’ (UDVS). A UDVS scheme can function as a standard publicly-verifiable digital signature but has additional functionality which allows *any* holder of a signature (not necessarily the signer) to *designate* the signature to any desired *designated-verifier* (using the verifier’s public key). Given the designated-signature, the designated-verifier can verify that the message was signed by the signer, but is unable to convince anyone else of this fact.

We propose an efficient *deterministic* UDVS scheme constructed using any bilinear group-pair. Our UDVS scheme functions as a standard Boneh-Lynn-Shacham (BLS) signature when no verifier-designation is performed, and is therefore compatible with the key-generation, signing and verifying algorithms of the BLS scheme. We prove that our UDVS scheme is secure in the sense of our unforgeability and privacy notions for UDVS schemes, under the Bilinear Diffie-Hellman (BDH) assumption for the underlying group-pair, in the random-oracle model. We also demonstrate a general constructive equivalence between a class of unforgeable and unconditionally-private UDVS schemes having unique signatures (which includes the *deterministic* UDVS schemes) and a class of ID-Based Encryption (IBE) schemes which contains the Boneh-Franklin IBE scheme but not the Cocks IBE scheme.

1 Introduction

In the modern world, one can find many examples of *user certification* systems. In these systems, a trusted *Certification Authority* (CA) issues signed certificates to *users*. Typically, the signed certificate attests to the truth of certain statements and attributes linked to the identity of the user to which the certificate is issued. A user Alice can present her certificate to any interested verifier Bob, who can in turn verify the CA’s signature and become convinced of the truth of the statements contained in the certificate. Real-life examples include the issuing of birth certificates, driving licences and academic transcripts.

In an electronic world, user certification systems can be implemented through the use of digital signatures. The ease of copying and transmitting electronic

certificates in such implementations is of great convenience to users; Alice can simply send a copy of her certificate to any interested verifier Bob. On the other hand, this same ease of distribution applies to Bob as well, who can easily disseminate Alice's certificate and convince an unlimited number of *third-party* verifiers about the truth of the statements concerning Alice contained in the certificate. This possibility poses a serious threat to Alice's *privacy*. Once Alice sends out her certificate to Bob she no longer has any control over the number of entities besides Bob who can not only learn all the statements about Alice contained in the certificate, but also become *convinced* about the truth of these statements by verifying the CA's signature on the certificate.

In this paper, we define a special type of digital signature scheme called a *Universal Designated-Verifier Signature* (UDVS) scheme, which directly addresses the above user privacy issue in user certification systems. Our scheme protects a user's privacy, and yet maintains a similar convenience of use for the user and for the certificate issuer CA as in certification systems using standard digital signatures. In a UDVS scheme, a user Alice is issued a signed certificate by the CA. When Alice wishes to send her certificate to a verifier Bob, she uses Bob's public key to transform the CA's signature into a *designated signature* for Bob, using the UDVS scheme's *designation* algorithm, and sends the certificate along with the *designated-signature* to Bob. Bob can use the CA's public key to verify the designated signature on the certificate, but is unable to use this designated signature to convince any other *third-party* that the certificate was signed by the CA, even if Bob is willing to reveal his secret-key to the *third-party*. This is achieved because Bob's secret-key allows him to forge designated signatures by himself, so the third-party is unable to tell who produced the signature (whereas Bob can, because he knows that he *didn't* produce it). Therefore, through the use of a UDVS scheme, the user Alice's privacy is preserved in the sense that Bob is unable to disseminate *convincing* statements about Alice (Of course, nothing prevents Bob from revealing the certificate statements themselves to any third-party, but the third-party will be unable to tell whether these statements are authentic, i.e. whether they have been signed by the CA or not).

We define quantitative notions of security for both the unforgeability and the privacy provided by UDVS schemes. We then propose an efficient UDVS scheme constructed from any bilinear group-pair, and we prove that this scheme satisfies our security requirements: it achieves perfect unconditional privacy and is unforgeable in the random-oracle model, assuming that the Bilinear Diffie-Hellman (BDH) assumption holds for the underlying bilinear group-pair. Our scheme has the attractive property that its signing, designation, and verification algorithms are all *deterministic*. We also show a general result which establishes a constructive equivalence between a class of unconditionally-private UDVS schemes possessing unique signatures (which contains all deterministic schemes) and a class of strongly-secure Identity-Based Encryption (IBE) schemes which contains the Boneh-Franklin IBE scheme [2], but not the Cocks IBE scheme [14]. Proofs of some statements have been omitted from the appendix due to lack of space.

They can be found in the full version of the paper uploaded to the IACR e-print archive.

1.1 Related Work

Our concept of UDVS schemes can be viewed as an application of the general idea of *designated-verifier proofs*, introduced by Jakobsson, Sako and Impagliazzo [21], where a prover non-interactively designates a proof of a statement to a verifier, in such a way that the verifier can simulate the proof by himself with his secret key and thus cannot transfer the proof to convince anyone else about the truth of the statement, yet the verifier himself is convinced by the proof. The authors of [21] also propose a *designated-verifier non-interactive undeniable signature*, in which the three-move zero-knowledge signature confirmation protocol of an undeniable signature [12] (converted to be non-interactive in the random-oracle model via the Fiat-Shamir heuristic [16]) is modified to be designated-verifier by replacing the commitment with a *trapdoor commitment* [7], in which the verifier's secret key is the trapdoor. However, the resulting scheme in [21] allows designation of signatures only by the *signer* (since designation requires the signer's secret key), whereas our UDVS scheme allows *anyone* who obtains a signature to designate it; this is what we mean by the term *universal* in the name 'Universal Designated-Verifier Signatures'. As we explain in Section 4.1, the idea in [21] of using a trapdoor commitment in a non-interactive zero-knowledge proof can also be used in principle to convert generic digital signature schemes into UDVS schemes. However, the use of a zero-knowledge proof results in a designation algorithm which is randomized, and typically inefficient. In contrast, we show that using bilinear group-pairs one can avoid zero-knowledge proofs and construct a UDVS scheme which has a deterministic and efficient designation algorithm.

There have been other approaches proposed to address the privacy threat associated with dissemination of verifiable signed documents. Chaum and van Antwerpen [10,12] introduced undeniable signatures for this purpose, which require a signer or confirmer's [13,27,25,9,17] interactive cooperation to verify a signature, but this approach places significant inconvenience and workload on verifiers and confirmers, compared to an off-line non-interactive verification. There has been substantial work on pseudonym-based *digital credentials* [11,6,5,8] which gives further approaches to enhance user privacy, such as *selective disclosure* of attributes (see also [31]) and *unlinkability* of user transactions. Chameleon signatures [24] allow designation of signatures to verifiers by the *signer*, and in addition allow a signer to prove a forgery by a designated verifier. Ring signatures [28], when restricted to two users, can also be viewed as designated-verifier signatures, where one user is the actual signer and the other user is the designated-verifier who can also forge the two-user ring signature, thus providing the privacy property, called *signer anonymity* in the context of ring signatures. However, signer designation is still performed by the *signer*. Recently, Boneh, Gentry, Lynn and Shacham [3] proposed a ring signature based on bilinear group-pairs and observed that it also allows public conversion of single-signer

ring signatures into two-signer ring signatures. Thus, the ring signature scheme in [3] can also be viewed as a UDVS scheme. However, we observe that our proposed UDVS scheme has two advantages over the UDVS scheme in [3]: (1) Our scheme has *deterministic* designation and signing algorithms, and therefore possesses *unique* designated-verifier signatures (unlike the randomized designation scheme in [3]). As we show in Sec. 5, secure UDVS schemes with unique signatures are as hard to construct as secure ID-Based Encryption (IBE) schemes (our scheme is related to the Boneh-Franklin IBE [2]), whereas this is not the case for randomized UDVS schemes, which can be constructed using other methods, (2) Our scheme extends the standard BLS signature [4], whereas the scheme in [3] is built on a modified BLS scheme. Our scheme also has an efficiency advantage in verification compared to [3] (see Section 6.1).

2 Preliminaries

2.1 Notation

We say that a function $f : \mathbb{N} \rightarrow \mathbb{R}$ is a *negligible* function if, for any $c > 0$, there exists $k_0 \in \mathbb{N}$ such that $f(k) < 1/k^c$ for all $k > k_0$. We say that a probability function $p : \mathbb{N} \rightarrow \mathbb{R}$ is *overwhelming* if the function $q : \mathbb{N} \rightarrow \mathbb{R}$ defined by $q(k) = 1 - p(k)$ is a negligible function. For various algorithms discussed, we will define a sequence of integers to measure the *resource parameters* of these algorithms (e.g. running-time plus program length, number of oracle queries to various oracles). All these resource parameters can in general be functions of a *security parameter* k of the scheme. We say that an algorithm A with resource parameters $RP = (r_1, \dots, r_n)$ is *efficient* if each resource parameter $r_i(k)$ of A is bounded by a polynomial function of the security parameter k , i.e. there exists a $k_0 > 0$ and $c > 0$ such that $r_i(k) < k^c$ for all $k > k_0$. For a probabilistic algorithm A , we use $A(x; r)$ to denote the output of A on input x with a randomness input r . If we do not specify r explicitly we do so with the understanding that r is chosen statistically independent of all other variables. We denote by $\{A(x)\}$ the set of outputs of A on input x as we sweep the randomness input for A through all possible strings.

2.2 Bilinear Group-Pairs

Our signature scheme proposed in Section 4.2 is built using a powerful cryptographic tool called a *Bilinear Group-Pair*. In this section we review the definition of a bilinear group-pair, following the definitions of [3]. We refer the reader to [22,23,2,4] for a discussion of how to build a concrete instance of such a group-pair using supersingular elliptic curves, and to [1] for efficient algorithms for computing the bilinear map over these group-pairs.

Definition 1 (Bilinear Group-Pair). *Let (G_1, G_2) denote a pair of groups of prime order $|G_1| = |G_2|$. We call the group-pair (G_1, G_2) a Bilinear Group-Pair if the pair (G_1, G_2) has the following properties:*

- (1) **Efficient Group Operations:** *The group operations in G_1 and G_2 are efficiently computable (in some representation).*
- (2) **Existence of Efficient Bilinear Map:** *There exists an efficiently computable bilinear map $e : G_1 \times G_2 \rightarrow G_T$ (for some image group G_T of order $|G_T| = |G_1| = |G_2|$) having the following properties:*
 - (a) **Bilinearity:** $e(u_1^{a_1}, u_2^{a_2}) = e(u_1, u_2)^{a_1 \cdot a_2}$ for all $(u_1, u_2) \in G_1 \times G_2$ and $(a_1, a_2) \in \mathbb{Z}^2$.
 - (b) **Non-Degeneracy:** $e(u_1, u_2) \neq 1$ for all $(u_1, u_2) \in G_1/\{1\} \times G_2/\{1\}$ (Here 1 denotes the identity element in the respective group).
- (3) **Existence of Efficient Isomorphism:** *There exists an efficiently computable group isomorphism $\psi : G_1 \rightarrow G_2$ from G_1 to G_2 .*

Our signature scheme's security relies on the computational hardness of the *Bilinear Diffie-Hellman* (BDH) problem associated with the bilinear group-pair used to construct the scheme. We review the BDH problem, and remark that the Boneh-Franklin ID-Based Encryption scheme [2] and Joux's tripartite key exchange protocol [22] also rely on the hardness of BDH.

Definition 2 (Bilinear Diffie-Hellman (BDH) Problem). *Let GC denote a randomized bilinear group-pair instance generation algorithm, which on input a security parameter k , outputs (D_G, g_1) , where $D_G \in \{0, 1\}^*$ is a description string for a bilinear group-pair (G_1, G_2) . We say that the BDH problem is hard in group-pairs generated by GC if, for any efficient attacker A, the probability $\text{Succ}_{A, \text{BDH}}(k)$ that A succeeds to compute $K = e(g_1, g_2)^{a \cdot b \cdot c}$ given $(D_G, g_1, g_1^a, g_1^b, g_2^c)$ for uniformly random $a, b, c \in \mathbb{Z}_{|G_1|}$, where $g_2 = \psi(g_1)$, is a negligible function of k (the probability is over A's random coins and the input to A). We quantify the insecurity of BDH against arbitrary attackers with running-time plus program length t by the probability $\text{InSec}_{\text{BDH}}(t) \stackrel{\text{def}}{=} \max_{A \in AS_{RP}} \text{Succ}_{A, \text{BDH}}(k)$, where the set AS_{RP} contains all attackers with run-time t .*

3 Universal Designated-Verifier Signature (UDVS) Schemes

3.1 Precise Definition of a UDVS Scheme

A Universal Designated Verifier Signature (UDVS) scheme DVS consists of seven algorithms and a 'Verifier Key-Registration Protocol' P_{KR} . All these algorithms may be randomized.

1. **Common Parameter Generation GC** — on input a security parameter k , outputs a string consisting of common scheme parameters cp (publicly shared by all users).
2. **Signer Key Generation GKS** — on input a common parameter string cp , outputs a secret/public key-pair (sk_1, pk_1) for *signer*.
3. **Verifier Key Generation GKV** — on input a common parameter string cp , outputs a secret/public key-pair (sk_3, pk_3) for *verifier*.

4. **Signing S** — on input signing secret key sk_1 , message m , outputs *signer's* publicly-verifiable (PV) signature σ .
5. **Public Verification V** — on input *signer's* public key pk_1 and message/PV-signature pair (m, σ) , outputs verification decision $d \in \{Acc, Rej\}$.
6. **Designation CDV** — on input a *signer's* public key pk_1 , a *verifier's* public key pk_3 and a message/PV-signature pair (m, σ) , outputs a designated-verifier (DV) signature $\hat{\sigma}$.
7. **Designated Verification VDV** — on input a *signer's* public key pk_1 , *verifier's* secret key sk_3 , and message/DV-signature pair $(m, \hat{\sigma})$, outputs verification decision $d \in \{Acc, Rej\}$.
8. **Verifier Key-Registration $P_{KR} = (KRA, VER)$** — a protocol between a ‘Key Registration Authority’ (KRA) and a ‘Verifier’ (VER) who wishes to register a verifier’s public key. On common input cp , the algorithms KRA and VER interact by sending messages alternately from one to another. At the end of the protocol, KRA outputs a pair $(pk_3, Auth)$, where pk_3 is a verifier’s public-key, and $Auth \in \{Acc, Rej\}$ is a key-registration authorization decision. We write $P_{KR}(KRA, VER) = (pk_3, Auth)$ to denote this protocol’s output.

Verifier Key-Reg. Protocol. The purpose of the ‘Verifier Key-Registration’ protocol is to force the verifier to ‘know’ the secret-key corresponding to his public-key, in order to enforce the non-transferability privacy property. In this paper we assume the *direct* key reg. protocol, in which the verifier simply reveals his key-pair (sk, pk) , and the KRA authorizes it only if $(sk, pk) \in \{GKV(cp)\}$ ¹.

Consistent UDVS Schemes. We require two obvious consistency properties from UDVS schemes. The ‘PV-Consistency’ property requires that the PV-signatures produced by the *signer* are accepted as valid by the PV-verification algorithm V. The ‘DV-Consistency’ property requires that the DV-signatures produced by the *designator* using the designation algorithm CDV are accepted as valid by the DV-verification algorithm VDV. We say that a UDVS scheme is *consistent* if it has both of the above consistency properties.

DVSig-Unique UDVS schemes. In this paper we are mainly interested in *DVSig-Unique* UDVS schemes, in which the DV signature $\sigma_{dv}^* = CDV(pk_1, pk_3, S(sk_1, m^*))$ on a message m^* by a signer with public key pk_1 (and secret key sk_1) to a verifier with public key pk_3 , is uniquely determined by (m^*, pk_1, pk_3) .

3.2 Security Properties of UDVS Schemes

3.2.1 Unforgeability. In the case of a UDVS scheme there are actually two types of unforgeability properties to consider. The first property, called ‘PV-Unforgeability’, is just the usual existential unforgeability notion under chosen-message attack [19] for the standard PV signature scheme $D = (GC, GKS, S, V)$ induced by the UDVS scheme (this prevents attacks to fool the

¹ The KRA can always perform this check efficiently, since we can assume that the secret key sk contains the randomness input to GKV used to generate it.

designator). The second property, called ‘DV-Unforgeability’, requires that it is difficult for an attacker to forge a DV-signature $\hat{\sigma}^*$ by the signer on a ‘new’ message m^* , such that the pair $(m^*, \hat{\sigma}^*)$ passes the DV-verification test with respect to a given designated-verifier’s public key pk_3 (this prevents attacks to fool the designated verifier, possibly mounted by a dishonest designator). It is easy to see that, due to the existence of the efficient public-designation algorithm CDV, the ‘DV-unforgeability’ property implies the ‘PV-unforgeability’ property², although the converse need not hold in general. Indeed, we will see that our proposed UDVS scheme’s ‘PV-unforgeability’ can be proven with a weaker assumption than that needed to prove the ‘DV-unforgeability’.

Definition 3 (DV-Unforgeability). Let $DVS = (GC, GKS, GKV, S, V, CDV, VDV, P_{KR})$ be a UDVS scheme. Let A denote a forger attacking the unforgeability of DVS. The DV-Unforgeability notion $UF\text{-}DV$ for this scheme is defined as follows:

1. **Attacker Input:** Signer and Verifier’s public-keys (pk_1, pk_3) (from $GKS(k), GKV(k)$).
2. **Attacker Resources:** Run-time plus program-length at most t , Oracle access to signer’s signing oracle $S(sk_1, \cdot)$ (q_s queries), and, if scheme DVS makes use of n random oracles RO_1, \dots, RO_n , allow q_{RO_i} queries to the i th oracle RO_i for $i = 1, \dots, n$. We write attacker’s Resource Parameters (RPs) as $RP = (t, q_s, q_{RO_1}, \dots, q_{RO_n})$.
3. **Attacker Goal:** Output a forgery message/DV-signature pair $(m^*, \hat{\sigma}^*)$ such that:
 - (1) The forgery is valid, i.e. $VDV(pk_1, sk_3, m^*, \hat{\sigma}^*) = \text{Acc}$.
 - (2) Message m^* is ‘new’, i.e. has not been queried by attacker to S .
4. **Security Notion Definition:** Scheme is said to be unforgeable in the sense of $UF\text{-}DV$ if, for any efficient attacker A , the probability $\text{Succ}_{A, DVS}^{UF\text{-}DV}(k)$ that A succeeds in achieving above goal is a negligible function of k . We quantify the insecurity of DVS in the sense of $UF\text{-}DV$ against arbitrary attackers with resource parameters $RP = (t, q_s, q_{RO_1}, \dots, q_{RO_n})$ by the probability

$$\text{InSec}_{DVS}^{UF\text{-}DV}(t, q_s, q_{RO_1}, \dots, q_{RO_n}) \stackrel{\text{def}}{=} \max_{A \in AS_{RP}} \text{Succ}_{A, DVS}^{UF\text{-}DV}(k),$$

where the set AS_{RP} contains all attackers with resource parameters RP .

3.2.2 Non-transferability Privacy. Informally, the purpose of the privacy property for a UDVS scheme is to prevent a designated-verifier from using the DV signature σ_{dv} on a message m to produce evidence which convinces a third-party that the message m was signed by the signer. Our model’s goal is to capture

² Actually, this assumes that $V(pk_1, m, \sigma) = \text{Acc}$ implies $\sigma \in \{S(sk_1, m)\}$ for all (m, σ) . But even if this does not hold, we can always redefine V to verify (m, σ) using pk_1 as follows: compute random key-pair $(sk_3, pk_3) = GKV(cp)$, compute $\hat{\sigma} = CDV(pk_1, pk_3, m, \sigma)$ and return $VDV(pk_1, sk_3, m, \hat{\sigma})$. It is easy to see that using this V , DV-Unforgeability implies PV-Unforgeability.

a setting in which signature holder provides many designated-signatures on m , designated to many verifier public keys of the attacker's choice. We quantify this property using the following privacy attack model. In our model, the attacker is a pair of interacting algorithms (A_1, A_2) representing the designated-verifier (DV) and Third-Party (TP), respectively, which run in two stages. At the end of Stage 1, A_1 decides on a message m^* to be signed by the signer. In Stage 2, A_1 obtains up to q_{d1} DV signatures $(\sigma_1, \dots, \sigma_{q_{d1}})$ by the signer on m^* from a designator oracle, designated to public-keys of A_1 's choice (these keys must first be registered by A_1 via key-reg. interactions with the KRA), and tries to use the σ_i 's to convince A_2 that the signer signed m^* . At the end of Stage 2, A_2 outputs an estimate $d \in \{\text{yes}, \text{no}\}$ for the answer to the question 'did the signer sign m^* '?

We associate with (A_1, A_2) a *convincing measure* $C_{\widehat{A}_1}(A_1, A_2)$ with respect to a forgery strategy \widehat{A}_1 , to measure the 'degree' to which A_2 can be convinced by A_1 that the signer signed m^* . It is defined as the *distinguisher advantage* of A_2 's estimate d to correctly distinguish between (1) The game **yes**, where the signer did sign m^* and A_1 obtained one or more DV signatures on m^* from the designator oracle or (2) The game **no**, where the signer did not sign m^* and A_1 was actually replaced by an efficient forging strategy, called \widehat{A}_1 (which accepts the program for A_1 as input), which aims to "trick" A_2 into believing that the signer signed m^* , without the need to obtain any DV signatures on m^* from the designator oracle. The scheme is said to achieve the privacy property if there is an efficient forgery strategy \widehat{A}_1 such that $C_{\widehat{A}_1}(A_1, A_2)$ is negligible for any efficient attacker pair (A_1, A_2) .

Definition 4 (PR-Privacy). Let $DVS = (GC, GKS, GKV, S, V, CDV, VDV, P_{KR})$ be a UDVS scheme. Let (A_1, A_2) denote an attack pair against the privacy of DVS. Let \widehat{A}_1 denote a forgery strategy. The privacy notion PR for this scheme is defined as follows:

1. **Attacker Input:** Signer public-key pk_1 (where $(sk_1, pk_1) = GKS(cp)$, and $cp = GC(k)$). Note that \widehat{A}_1 also accepts the program for A_1 as input.
2. **Resources for (A_1, \widehat{A}_1) :** Run-time (t_1, \widehat{t}_1) , access to signing oracle $S(sk_1, \cdot)$ (up to (q_s, \widehat{q}_s) queried messages different from m^*), access to key-reg. protocol interactions with the KRA (up to (q_k, \widehat{q}_k) interactions), access to A_2 oracle (up to (q_c, \widehat{q}_c) messages). In stage 2, A_1 also has access to designation oracle $CDV(pk_1, \cdot, m^*, \sigma^*)$ (up to q_d queried keys successfully registered with KRA), where $\sigma^* = S(sk_1, m^*)$ is a signer's signature on the challenge message m^* output by A_1 at end of Stage 1. Note that \widehat{A}_1 cannot make any designation queries.
3. **Resources for A_2 :** Run-time t_2 .
4. **Attacker Goal:** Let $P(A_1, A_2)$ and $P(\widehat{A}_1, A_2)$ denote the probabilities that A_2 outputs **yes** when interacting with A_1 (game **yes**) and \widehat{A}_1 (game **no**), respectively. The goal of (A_1, A_2) is to achieve a non-negligible convincing measure $C_{\widehat{A}_1}(A_1, A_2) \stackrel{\text{def}}{=} |P(A_1, A_2) - P(\widehat{A}_1, A_2)|$.

5. Security Notion Definition: Scheme is said to achieve privacy in the sense of PR if there exists an efficient forgery strategy \widehat{A}_1 such that the convincing measure $C_{\widehat{A}_1}(A_1, A_2)$ achieved by any efficient attacker pair (A_1, A_2) is negligible in the security parameter k . We quantify the insecurity of DVS in the sense of PR against arbitrary attacker pairs (A_1, A_2) with resources (RP_1, RP_2) (attacker set AS_{RP_1, RP_2}), with respect to arbitrary forgery strategies \widehat{A}_1 with resources \widehat{RP}_1 (attacker set $AS_{\widehat{RP}_1}$) by the probability

$$\text{InSec}_{\text{DVS}}^{PR}(RP_1, \widehat{RP}_1, RP_2) \stackrel{\text{def}}{=} \min_{\widehat{A}_1 \in AS_{\widehat{RP}_1}} \max_{(A_1, A_2) \in AS_{RP_1, RP_2}} C_{\widehat{A}_1}(A_1, A_2).$$

If $\text{InSec}_{\text{DVS}}^{PR}(RP_1, \widehat{RP}_1, RP_2) = 0$ holds for any computationally unbounded A_2 , it is said to be perfect unconditional privacy. If privacy holds when $q_{s1} = q_{s1}$ it is said to be complete privacy.

Remark. The above privacy notion handles general UDVS schemes. For more specific schemes, the definition can be simplified. For instance, for schemes using the *direct* key-reg. protocol which have unique signatures, the complete unconditional privacy is equivalent to the existence of an efficient universal forgery algorithm for DV signatures using the verifier's secret key (this is the case for our proposed scheme in this paper, but see Sec. 6.2 for other possibilities).

Lemma 1. Let $\text{DVS} = (\text{GC}, \text{GKS}, \text{GKV}, \text{S}, \text{V}, \text{CDV}, \text{VDV}, P_{KR})$ be a UDVS scheme which is DV-Sig-Unique, and where P_{KR} is the direct key-reg. protocol. Then DVS achieves complete and perfect unconditional privacy if and only if there exists an efficient universal DV-sig. forgery algorithm F , which on any input $(cp, pk_1, sk_3, pk_3, m^*)$ (where $(sk_1, pk_1) \in \{\text{GKS}(cp)\}$ and $(sk_3, pk_3) \in \{\text{GKV}(cp)\}$) computes the unique DV-sig. $\sigma_{dv}^* = \text{CDV}(cp, pk_1, pk_3, \text{S}(sk_1, m^*))$ with probability 1.

4 Proposed UDVS Scheme

4.1 An Inefficient Generic Approach for Constructing UDVS Schemes

Before we present our efficient UDVS scheme, we sketch, as a plausibility argument, the details of a generic (but inefficient) approach for constructing UDVS schemes, based on zero-knowledge designated-verifier proofs of membership [21]. We do not attempt to give a precise definition and proof of security properties for this generic scheme, but we believe this can be done along the outline we sketch below.

The generic construction works as follows. We make use of a standard digital signature scheme $\text{DS} = (\text{GK}_S, \text{S}, \text{V})$ which is secure in the standard CMA sense of existential unforgeability under chosen message attack [19]. We also need a public-key encryption scheme $\text{PKE} = (\text{GK}_E, \text{E}, \text{D})$ which is semantically secure under chosen-plaintext attack (IND-CPA) [20], and a trapdoor commitment

scheme TC [7]. The common parameter generation algorithm GC for the UDVS scheme generates an encryption key-pair $(sk_E, pk_E) = \text{GK}_E(k)$ and outputs pk_E as the common parameter. The signer key-generation/signing/PV-verification algorithms for the UDVS scheme are those of the signature scheme DS. The verifier's key-generation algorithm is that of the trapdoor commitment scheme TC. The designation algorithm CDV takes an input common parameter pk_E , message m and its signature σ_{pv} , signer's public key pk_1 , and verifier's public key pk_3 . The designated signature σ_{dv} is the pair (c, P) , where $c = \text{E}(pk_E, \sigma_{pv}; r)$ is the encryption of σ_{pv} under the common public key pk_E (using a random string r), and P is a designated-verifier non-interactive proof that c is in the NP language

$$L_{pk_1, pk_E, m} = \{c : \exists(\sigma, r) \text{ such that } c = \text{E}(pk_E, \sigma; r) \text{ and } \text{V}(pk_1, m, \sigma) = \text{Acc}\},$$

consisting of all possible ciphertexts of valid signatures by the signer on the message m . Note that the designator has a witness (σ, r) for membership of c in $L_{pk_1, pk_E, m}$, and hence can use a generic zero-knowledge commit-challenge-response proof of membership for NP languages [18] to prove that $c \in L_{pk_1, pk_E, m}$. By applying the Fiat-Shamir heuristic (replacing the challenge by a hash of the commitments) to make the proof non-interactive and using the verifier's trapdoor commitment in the commit step, the designator can compute the desired designated-verifier proof P . The DV verification algorithm consists of verifying the proof P for the ciphertext c . The DV-unforgeability of the scheme follows (in the random oracle model) from the soundness of the proof and the unforgeability of the underlying standard signature scheme: any forged ciphertext with a valid proof is by soundness a ciphertext of a valid signature and can be decrypted with sk_E to give a forgery for the underlying signature scheme. The (computational) privacy follows from the forgeability of the proof P by the designated-verifier using his secret-key, namely the commitment trapdoor (even for ciphertexts c not in the language $L_{pk_1, pk_E, m}$), and the (computational) simulatability of the ciphertext c by a random string, due to the semantic security of the encryption scheme.

Implementing the above scheme using a generic zero-knowledge NP proof system [18] would yield a very inefficient and randomized designation and inefficient DV verification. For specific choices of the underlying signature and encryption scheme, one may be able to give a more efficient zero-knowledge proof for the language $L_{pk_1, pk_E, m}$ and improve this efficiency to some extent. However, our bilinear scheme below shows how to eliminate zero-knowledge proofs altogether and obtain efficient, deterministic designation.

4.2 An Efficient UDVS Scheme DVSBM Based on Bilinear Group-Pairs

Our proposed UDVS scheme DVSBM based on bilinear group-pairs is given below. It makes use of a cryptographic hash function $H : \{0, 1\}^{\leq \ell} \rightarrow G_2$, modelled as a random-oracle. Here ℓ denotes a bound on the message bit-length and $\{0, 1\}^{\leq \ell}$ denotes the message space of all strings of length at most ℓ bits.

Note that for the basic version of DVSBM we propose the *direct* key registration protocol (see Section 3.1).

1. **Common Parameter Generation GC.** Choose a bilinear group-pair (G_1, G_2) of prime order $|G_1| = |G_2|$ with description string D_G specifying a bilinear map $e : G_1 \times G_2 \rightarrow G_T$, isomorphism $\psi : G_1 \rightarrow G_2$ and generators $g_1 \in G_1$ and $g_2 = \psi(g_1) \in G_2$. The common parameters are $cp = (D_G, g_1)$.
2. **Signer Key Generation GKS.** Given cp , pick random $x_1 \in \mathbb{Z}_{|G_1|}$ compute $y_1 = g_1^{x_1}$. The public key is $pk_1 = (cp, y_1)$. The secret key is $sk_1 = (cp, x_1)$.
3. **Verifier Key Generation GKV.** Given cp , pick random $x_3 \in \mathbb{Z}_{|G_1|}$ compute $y_3 = g_1^{x_3}$. The public key is $pk_3 = (cp, y_3)$. The secret key is $sk_3 = (cp, x_3)$.
4. **Signing S.** Given the signer's secret key (cp, x_1) , and message m , compute $\sigma = h^{x_1} \in G_2$, where $h = H(m)$. The PV signature is σ .
5. **Public Verification V.** Given the signer's public key (cp, y_1) and a message/PV-sig. pair (m, σ) , accept if and only if $e(g_1, \sigma) = e(y_1, h)$, where $h = H(m)$.
6. **Designation CDV.** Given the signer's public key (cp, y_1) , a verifier's public key (cp, y_3) and a message/PV-signature pair (m, σ) , compute $\hat{\sigma} = e(y_3, \sigma)$. The DV signature is $\hat{\sigma}$.
7. **Designated Verification VDV.** Given a signer's public key (cp, y_1) , a verifier's secret key (cp, x_3) , and message/DV-sig. pair $(m, \hat{\sigma})$, accept if and only if $\hat{\sigma} = e(y_1^{x_3}, h)$, where $h = H(m)$.

Consistency. We first demonstrate the consistency of scheme DVSBM. To show the PV-Consistency property, we note that if $\sigma_{pv} \stackrel{\text{def}}{=} S(sk_1, m) = h^{x_1}$, where $h = H(m)$, then

$$e(g_1, \sigma) = e(g_1, h^{x_1}) = e(g_1, h)^{x_1} = e(g_1^{x_1}, h) = e(y_1, h) \quad (1)$$

by bilinearity, so $V(pk_1, m, \sigma_{pv}) = \text{Acc}$, as required. To show the DV-Consistency property, we note that if $\sigma_{pv} \stackrel{\text{def}}{=} S(sk_1, m) = h^{x_1}$, where $h = H(m)$, then $\sigma_{dv} \stackrel{\text{def}}{=} \text{CDV}(pk_1, pk_3, m, \sigma_{pv}) = e(y_3, \sigma_{pv})$. Consequently:

$$\hat{\sigma}_{dv} \stackrel{\text{def}}{=} e(y_1^{x_3}, h) = e(y_3^{x_1}, h) = e(y_3, h)^{x_1} = e(y_3, h^{x_1}) = e(y_3, \sigma_{pv}) = \sigma_{dv} \quad (2)$$

by bilinearity, so $\text{VDV}(pk_1, (sk_3, pk_3), m, \sigma_{dv}) = \text{Acc}$, as required. Therefore the scheme DVSBM is *consistent*.

Unforgeability. In the random-oracle model for $H(\cdot)$, we can prove the DV-unforgeability of the scheme assuming the Bilinear Diffie-Hellman (BDH) assumption. The reduction is efficient (no q_H multiplicative cost in insecurity bound) thanks to the random self-reducibility of BDH, by adapting Coron's technique [15] which was originally applied to prove the unforgeability of the FDH – RSA signature scheme assuming the RSA assumption. We note that the PV-unforgeability of our scheme reduces to the unforgeability of the BLS scheme [4], which was proven in [4] under a weaker assumption than hardness of BDH, namely hardness of the 'co-CDH' assumption.

Theorem 1 (DV-unforgeability of DVSBM). *If the Bilinear Diffie-Hellman problem is hard in the bilinear group-pairs (G_1, G_2) generated by the common-parameter algorithm GC, then the scheme DVSBM is DV-unforgeable (UF-DV notion) in the random-oracle model for $H(\cdot)$. Concretely, the following insecurity bound holds:*

$$\mathbf{InSec}_{\text{DVSBM}}^{\text{UF-DV}}(t, q_s, q_H) \leq \exp(1) \cdot (q_s + 1) \cdot \mathbf{InSec}_{\text{BDH}}(t[B]), \quad (3)$$

where $t[B] = t + (q + q_s + 1) \cdot O(\ell \cdot \log_2 q + T_g \cdot \log_2 |G_1|) + T_\psi + T_e$. Here we define $q \stackrel{\text{def}}{=} q_H + q_s + 1$ and denote by T_e , T_g , and T_ψ the running time bounds for evaluating the bilinear map e , performing a single group operation in G_1 or G_2 , and evaluating the isomorphism ψ , respectively, and we use $\exp : \mathbb{R} \rightarrow \mathbb{R}$ to denote the natural exponential function.

Privacy. The privacy achieved by scheme DVSBM is perfect unconditional, because the verifier can easily forge the DV-signatures he is receiving from the designator (as long as the verifier knows his secret-key, which is ensured by the key-registration protocol).

Theorem 2 (Privacy of DVSBM). *The scheme DVSBM achieves complete and perfect unconditional privacy (in the sense of the PR notion). Concretely:*

$$\mathbf{InSec}_{\text{DVSBM}}^{\text{PR}}(RP_1, \widehat{RP}_1, \infty) = 0, \quad (4)$$

where $RP_1 = (t_1, q_s, q_k, q_d)$ denotes A_1 's resource parameters and $\widehat{RP}_1 = (\widehat{t}_1, \widehat{q}_s, \widehat{q}_k)$ denotes the forgery strategy \widehat{A}_1 's resources, which are given by: $\widehat{t}_1 = t_1 + q_d \cdot O(T_e + T_g \log_2 |G_1| + q_k)$, $\widehat{q}_s = q_s$ (complete privacy), $\widehat{q}_d = q_d$, $\widehat{q}_c = q_c$.

5 General Relationship between UDVS and ID-Based Encryption Schemes

Readers who are familiar with the Boneh-Franklin ID-Based Encryption scheme [2] may notice an intimate relationship between that scheme and our proposed UDVS scheme DVSBM. In this section we show that this relationship is one instance of a general equivalence between certain subclass of secure UDVS schemes and a certain subclass of secure ID-Based Encryption schemes.

ID-Based Key Encapsulation Mechanism (ID-KEM) Schemes. We review the definition of ID-based encryption schemes (IBE) [2]. Actually, we will formulate our result in terms of a primitive called ‘ID-Based Key Encapsulation Mechanism’ (ID-KEM), defined analogously to the definition of standard non-ID-based KEMs [30]. An ID-Based Key Encapsulation Mechanism (ID-KEM) consists of 4 algorithms: **Setup**, **Extract**, **Encrypt**, **Decrypt**: **Setup** takes as input security parameter k , and returns a system parameter string cp and a master key mk . This is run initially by the ‘Private Key Generator’ (PKG). **Extract** takes as input system parameters cp , master key mk , and a user identity string $ID \in S_{ID}$ and returns a user secret key $sk_{ID} = \text{Extract}(cp, mk, ID)$ corresponding to identity

ID. Encrypt is a randomized algorithm which takes as input system parameters cp , a recipient identity string ID and a random input $r \in S_R$ and returns a pair $(K, c) = \text{Encrypt}(cp, ID; r)$, where $K = \text{Enc}_K(cp, ID; r)$ is an ‘session key’ (which can be used with a symmetric encryption scheme to encrypt a message) and $c = \text{Enc}_c(cp, ID; r)$ is a ciphertext for K (we call Enc_K and Enc_c the *key-computation* and *key-encapsulation* functions induced by *Encrypt*). Given cp , sk_{ID} and c , $\text{Decrypt}(cp, sk_{ID}, c)$ recovers a session key K . An ID-KEM is *consistent* if $\text{Decrypt}(cp, sk_{ID}, \text{Enc}_c(cp, ID; r)) = \text{Enc}_K(cp, ID; r)$ holds, where $sk_{ID} = \text{Extract}(cp, mk, ID)$, for all (ID, r) and (cp, mk) generated by *Setup*.

Ephemeral-Key (EK) and Separable ID-KEM Schemes. For constructing UDVS schemes, we need an ID-KEM scheme which satisfies two properties: EK and Separable. An ID-KEM scheme is said to have the *EK* property if the ciphertext $c = \text{Enc}_c(cp, ID; r)$ produced by the key-encapsulation function Enc_c does not depend on ID . An ID-KEM scheme is said to be *Separable* if the *Setup* can be separated into two efficient algorithms Setup_1 and Setup_2 such that the following holds. On input security parameter k , $\text{Setup}_1(k)$ returns a string cp_1 , and on input cp_1 , $\text{Setup}_2(cp_1)$ returns a master key mk and a second string cp_2 . The system parameter string is $cp = (cp_1, cp_2)$, and we require that the ciphertext $c = \text{Enc}_c((cp_1, cp_2), ID; r)$ produced by the key-encapsulation function Enc_c does not depend on cp_2 .

Strong ID-One-Wayness. Following the definition in [2], a basic security requirement for ID-KEM schemes is *ID-One-Wayness* (ID-OW). For constructing UDVS schemes, we need a stronger requirement that we call *Strong ID-One-Wayness* (ST-ID-OW). An ID-KEM scheme is said to have the ST-ID-OW property if it is infeasible for an attacker A to win the following two-stage game. In Stage 1, A is given the system pars. cp and outputs a recipient identity ID he wants to be challenged on. In Stage 2, A is given a random KEM challenge ciphertext $c = \text{Enc}_c(cp, ID; r)$ intended for recipient ID but we allow A to adaptively ‘change his mind’ about the challenge identity; at the end of Stage 2, A outputs an identity ID^* and an estimate \hat{K}^* for the decryption $K^* = \text{Decrypt}(cp, sk_{ID^*}, c)$ of c under secret-key sk_{ID^*} corresponding to identity ID^* . A is said to win if $\hat{K}^* = K^*$ (in both stages, A is allowed to query any $ID' \neq ID^*$ to the *Extract* oracle). Note that in the weaker ID-OW notion [2] A is not able to change the identity picked at the end of Stage 1.

We remark that the Boneh-Franklin IBE [2] can be seen as derived from an underlying separable EK ID-KEM, whereas the Cocks IBE scheme [14] does not seem to give rise to such a KEM.

Constructing a UDVS Scheme from a Separable EK ID-KEM Scheme. We can now describe our general construction of a UDVS scheme from a Separable EK ID-KEM scheme which achieves strong ID-OneWayness.

1. **Com. Par. Generation GC.** Compute $cp_1 = \text{Setup}_1(k)$. The common parameters are cp_1 .
2. **Signer Key Generation GKS.** Given common parameters cp_1 , compute $(cp_2, mk) = \text{Setup}_2(cp_1)$. The public key is (cp_1, cp_2) . The secret key is (cp_1, cp_2, mk) .

3. **Verifier Key Generation GKV.** Given common parameters cp_1 , let ID_0 and cp_{20} denote any fixed strings. Compute KEM ciphertext $c = \text{Enc}_c((cp_1, cp_{20}), ID_0; r_c)$ for uniformly random $r_c \in S_R$. The public key is c . The secret key is r_c .
4. **Signing S.** Given the signer's secret key (cp_1, cp_2, mk) , and message m , compute $sk_m = \text{Extract}((cp_1, cp_2), mk, m)$. The PV signature is sk_m .
5. **Public Verification V.** Given the signer's public key (cp_1, cp_2) and a message/PV-sig. pair (m, sk_m) , compute a random KEM ciphertext to identity string m as $\hat{c} = \text{Enc}_c((cp_1, cp_2), m; \hat{r})$ for uniformly random $\hat{r} \in S_R$ with associated encapsulated key $\hat{K} = \text{Enc}_K((cp_1, cp_2), m; \hat{r})$. Accept if and only if $\hat{K}' = \hat{K}$, where $\hat{K}' = \text{Decrypt}((cp_1, cp_2), sk_m, \hat{c})$.
6. **Designation CDV.** Given the signer's public key (cp_1, cp_2) , verifier's public key c and a message/PV-sig. pair (m, sk_m) , compute $K_{c,m} = \text{Decrypt}((cp_1, cp_2), sk_m, c)$. The DV signature is $K_{c,m}$.
7. **Designated Verification VDV.** Given a signer's public key (cp_1, cp_2) , a verifier's secret key r_c , and message/DV-sig. pair $(m, K_{c,m})$, compute $\hat{K}_{c,m} = \text{Enc}_K((cp_1, cp_2), m; r_c)$ and accept if and only if $\hat{K}_{c,m} = K_{c,m}$.

The underlying idea behind the construction is a correspondence between the ID-KEM setting and the UDVS setting, where one can make associations between: signer and PKG, messages and identities, DV-sigs. and session keys, designator and decryptor, verifier and encryptor. We point out however the reasons behind the necessity of the special requirements on the ID-KEM scheme: (1) The DV-Consistency of the UDVS scheme translates to the requirement on the ID-KEM scheme that if $c = \text{Enc}_c((cp_1, cp_{20}), ID_0; r_c)$ then $\text{Decrypt}((cp_1, cp_2), sk_{ID}, c) = \text{Enc}_K((cp_1, cp_2), ID; r)$ for *any* ID and the corresponding secret key sk_{ID} to ID . This requirement is satisfied by all Separable EK ID-KEM schemes, but not for general ID-KEM schemes. (2) The ID-KEM separability property is necessary in order that the verifier key-generation algorithm GKV does not need the signer's public key pk_1 — we require a UDVS scheme to allow verifiers to generate keys just once, not once per signer. (3) The ID-KEM needs to have the *strong* ID-OneWayness to ensure the *existential* DV unforgeability for the constructed UDVS scheme.

Constructing an ID-KEM from a UDVS scheme. Interestingly, the above correspondence can also be used in the other direction to construct an ID-KEM scheme (and hence an IBE scheme) from any DV-unforgeable UDVS scheme which is DV-Sig-Unique and achieves perfect unconditional privacy. The latter properties are needed for the consistency of the ID-KEM construction. The ID-KEM construction is as follows (we let F denote the universal forgery algorithm associated with the UDVS scheme, which exists by Lemma 1).

1. **System Par. Gen. Setup.** Given security parameter k , compute $cp = \text{GC}(k)$ and $(sk_1, pk_1) = \text{GKS}(cp)$. The system parameters are (cp, pk_1) . The master key is (cp, sk_1) .
2. **Secret-Key Extraction Extract.** Given master key sk_1 and identity ID , compute $\sigma_{ID} = S(sk_1, ID)$. The identity secret key is σ_{ID} .

3. **KEM Encryption Encrypt.** Given system par. (cp, pk_1) , identity ID and random input r , compute $(sk_3, pk_3) = \text{GKV}(cp, r)$ using random input r and DV-sig. forgery $\hat{\sigma}_{ID, pk_3} = \text{F}(cp, pk_1, sk_3, pk_3, ID)$. The KEM ciphertext is pk_3 . The encapsulated key is $\hat{\sigma}_{ID, pk_3}$.
4. **Decryption Decrypt.** Given system par. (cp, pk_1) , secret key σ_{ID} corresponding to identity ID , and KEM ciphertext pk_3 , compute DV-sig. $\hat{\sigma}'_{ID, pk_3} = \text{CDV}(pk_1, pk_3, ID, \sigma_{ID})$. The decrypted encapsulated key is $\hat{\sigma}'_{ID, pk_3}$.

We summarise our equivalence result in the following statement.

Theorem 3 (Equivalence between subclasses of ID-KEM and UDVS Schemes). (1) *Given any separable and EK ID-KEM scheme $\text{KEM} = (\text{Setup}_1, \text{Setup}_2, \text{Extract}, \text{Encrypt}, \text{Decrypt})$ which is consistent and achieves Strong ID-One-Wayness (ST-ID-OW notion), we can construct a UDVS scheme which is consistent and DVSig-Unique and achieves complete perfect unconditional privacy (PR notion) and DV-unforgeability (UF-DV notion).*

(2) *Conversely, given any UDVS scheme $\text{DVS} = (\text{GC}, \text{GKS}, \text{GKV}, \text{S}, \text{V}, \text{CDV}, \text{VDV}, \text{PKR})$ (where PKR is the direct key-reg. protocol) which is DVSig-Unique, consistent, and achieves complete perfect unconditional privacy (PR notion) and DV-unforgeability (UF-DV notion), we can construct an EK ID-KEM scheme KEM which is consistent and achieves Strong ID-One-Wayness (ST-ID-OW notion).*

6 Implementation Aspects and Extensions

6.1 Practical Efficiency of UDVS Scheme DVSBM

Currently, the only known way to construct bilinear group-pairs in which BDH is hard is to set G_1 and G_2 to be subgroups of the group of points on certain elliptic curves, as described in [23,2,4,1]. As shown in [1], for such implementations it is possible to evaluate the bilinear map in less than 20ms on a 1GHz P-III processor. Thus we believe that such potential implementations of our scheme are quite practical for many applications of UDVS schemes. Compared to the ring signature in [3], which can also function as a UDVS scheme when restricted to Two-Users as mentioned in Section 1.1, our scheme requires only a single pairing evaluation for designated verification (plus an exponentiation) whereas the scheme in [3] requires three pairing evaluations for this purpose. On the other hand, the scheme in [3] requires only two exponentiations for designation, which may be more efficient than the one pairing evaluation for designation in our scheme.

6.2 Achieving Unforgeability against the KRA

One may require unforgeability of DV-sigs. even against the KRA, which is a stronger than DV-unforgeability notion we defined. To achieve this one can replace the direct key-reg. protocol that we assumed by a zero-knowledge proof

of knowledge of the verifier's secret-key. For the scheme DVSBM, the Schnorr proof of knowledge of discrete-logs protocol [29] should suffice for this purpose, although we do not claim a formal proof of security for the resulting scheme.

6.3 Communication-Efficient Selective Disclosure for UDVS Scheme DVSBM

In the application of UDVS schemes to certification systems, Alice's certificate may contain n statements, but Alice may wish to further protect her privacy by disclosing only a *selected subset* of $r < n$ signed statements to Bob. This is easily achieved if Alice obtains a separate signature from the CA for each statement, but requires Alice to send (and designate) r signatures to Bob. Here we observe that for the scheme DVSBM, Alice can reduce the communication cost to only a single DV signature length (and also reduce her computation cost to only one designation and $r - 1$ group operations) by using similar techniques as used in [3]. Namely, given the PV signatures $(\sigma_1, \dots, \sigma_r)$ by a signer with public key $y_1 = g_1^{x_1}$ on messages (m_1, \dots, m_r) , with $\sigma_i = h_i^{x_1}$ and $h_i = H(m_i)$ for $i = 1, \dots, r$, a user who wishes to designate a signature on these messages to a verifier with public key $y_3 = g_1^{x_3}$, first multiplies the PV signatures to get $\sigma = \sigma_1 \cdots \sigma_r$, and then designates the product to get $\sigma_{dv} = e(y_3, \sigma)$. The verifier receives (m_1, \dots, m_r) , y_1 and σ_{dv} , computes $\hat{\sigma}_{dv} = e(y_1^{x_3}, h)$ where $h = h_1 \cdots h_r$ and checks that $\hat{\sigma}_{dv} = \sigma_{dv}$. The scheme can be proved DV-unforgeable in the 'aggregate signature' sense defined in [4] by reduction from the DV-unforgeability of DVSBM.

7 Conclusions and Future Work

We introduced *Universal Designated-Verifier Signature* (UDVS) schemes to improve the privacy of users in certification systems while maintaining the ease of use of electronic certificates. We defined precise security notions for UDVS schemes, proposed an efficient deterministic UDVS scheme based on bilinear group-pairs, and proved that our scheme achieves our desired security notions (in the random-oracle model), assuming the hardness of the Bilinear Diffie Hellman problem for the underlying group-pair. We also showed a general relationship between UDVS schemes and ID-Based encryption schemes, and discussed extensions to our basic scheme. In [26], we extend this work and show how to construct practical randomized UDVS schemes based on the classical Diffie-Hellman and RSA problems, in the random-oracle model. Threshold versions of UDVS schemes, in which the designator or designated-verifier consist of groups of users, are an interesting topic for future research. Another interesting problem is to construct a practical UDVS scheme which is unforgeable in the *standard* computational model with respect to established cryptographic assumptions.

Acknowledgments

The work of Ron Steinfeld, Huaxiong Wang and Josef Pieprzyk was supported by ARC Discovery Grant DP0345366. Huaxiong Wang's work was also supported in part by ARC Discovery Grant DP0344444.

References

1. P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient Algorithms for Pairing-based Cryptosystems. In *Crypto 2002*, volume 2442 of *LNCS*, pages 354–368, Berlin, 2002. Springer-Verlag.
2. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229, Berlin, 2001. Springer-Verlag.
3. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Eurocrypt 2003*, *LNCS*, Berlin, 2003. Springer-Verlag. (To Appear).
4. D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–532, Berlin, 2001. Springer-Verlag. See full updated version available at <http://crypto.stanford.edu/~dabo/pubs.html>.
5. S. Brands. A technical overview of digital credentials, 1999. Available at <http://www.xs4all.nl/~brands/>.
6. S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates*. MIT Press, 2000.
7. Gilles Brassard, David Chaum, and Claude Cr  peau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, October 1988.
8. J. Camenisch and A. Lysyanskaya. An Efficient System for Non-Transferrable Anonymous Credentials with Optional Anonymity Revocation. In *Eurocrypt 2001*, volume 2045 of *LNCS*, pages 93–118, Berlin, 2003. Springer-Verlag.
9. J. Camenisch and M. Michels. Confirmer Signature Schemes Secure against Adaptive Adversaries. In *Eurocrypt 2000*, volume 1807 of *LNCS*, pages 243–258, Berlin, 2000. Springer-Verlag.
10. D. Chaum and H. van Antwerpen. Undeniable Signatures. In *Crypto '89*, volume 435 of *LNCS*, pages 212–216, Berlin, 1990. Springer-Verlag.
11. D. Chaum. Security without ID: Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.
12. D. Chaum. Zero-Knowledge Undeniable Signatures. In *Eurocrypt '90*, volume 473 of *LNCS*, pages 458–464, Berlin, 1991. Springer-Verlag.
13. D. Chaum. Designated Confirmer Signatures. In *Eurocrypt '94*, volume 950 of *LNCS*, pages 86–91, Berlin, 1994. Springer-Verlag.
14. C. Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In *Cryptography and Coding 2001*, volume 2260 of *LNCS*, pages 360–363, Berlin, 2001. Springer-Verlag.
15. J-S. Coron. On the Exact Security of Full Domain Hash. In *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235, Berlin, 2000. Springer-Verlag.
16. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Berlin, 1987. Springer-Verlag.
17. R. Gennaro and T. Rabin. RSA-Based Undeniable Signatures. *J. of Cryptology*, 13:397–416, 2000.
18. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):690–728, 1991.

19. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure against Adaptively Chosen Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
20. S. Goldwasser and S. Micali. Probabilistic Encryption. *J. of Computer and System Sciences*, 28(2):270–299, 1984.
21. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and Their Applications. In *Eurocrypt '96*, volume 1070 of *LNCS*, pages 143–154, Berlin, 1996. Springer-Verlag.
22. A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Fourth Algorithmic Number Theory Symposium (ANTS IV)*, volume 1838 of *LNCS*, pages 385–394, Berlin, 2000. Springer-Verlag.
23. A. Joux. The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. In *Fifth Algorithmic Number Theory Symposium (ANTS V)*, volume 2369 of *LNCS*, pages 20–32, Berlin, 2002. Springer-Verlag.
24. H. Krawczyk and T. Rabin. Chameleon Signatures. In *NDSS 2000*, 2000. Available at <http://www.isoc.org/isoc/conferences/ndss/2000/proceedings/>.
25. M. Michels and M. Stadler. Generic Constructions for Secure and Efficient Confirmer Signature Schemes. In *Eurocrypt '98*, volume 1403 of *LNCS*, pages 406–421, Berlin, 1998. Springer-Verlag.
26. Authors of this paper, 2003. Work in progress.
27. T. Okamoto. Designated Confirmer Signatures and Public-Key Encryption are Equivalent. In *Crypto '94*, volume 839 of *LNCS*, pages 61–74, Berlin, 1994. Springer-Verlag.
28. R. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. In *Asiacrypt 2001*, volume 2248 of *LNCS*, pages 552–565, Berlin, 2001. Springer-Verlag.
29. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO'89*, volume 435 of *LNCS*, pages 239–251, Berlin, 1990. Springer-Verlag.
30. V. Shoup. *A Proposal for an ISO Standard for Public Key Encryption (Version 1.1)*, December 2001. ISO/IEC JTC 1/SC 27.
31. R. Steinfeld, L. Bull, and Y. Zheng. Content Extraction Signatures. In *International Conference on Information Security and Cryptology ICISC 2001*, volume 2288 of *LNCS*, pages 285–304, Berlin, 2001. Springer-Verlag.

A Proofs

A.1 Proof of Theorem 2

We first observe that DVSBM is a DVSig-Unique scheme. This follows immediately from the facts that there is only one secret key $x_1 \in \mathbb{Z}_{|G_1|}$ corresponding to each signer public key $y_1 = g_1^{x_1}$ (since g_1 is a generator), and the signing and designation algorithms are both deterministic. Secondly, we observe that there exists an efficient universal DV signature forgery algorithm F , which on input $(cp, y_1, (x_3, y_3), m^*)$, computes the unique DV signature $\sigma_{dv} = \text{CDV}(cp, y_1, y_3, (x_3, y_3), m^*, S(cp, x_1, m^*))$ with probability 1. Namely, F simply computes $\hat{\sigma} = e(y_1^{x_3}, h)$ as done by the DV verification algorithm, which is equal to σ_{dv} , by the DV-Consistency property Eq. (2). The algorithm F runs in time $t_F = O(T_g \cdot \log_2 |G_1|) + T_e$. We now construct the forgery strategy \widehat{A}_1 as in the

proof of Lemma 1, where \widehat{A}_1 simply runs A_1 and perfectly simulates its designation queries using F and the appropriate verifier secret keys from corresponding KRA queries of A_1 . The run-time of \widehat{A}_1 is the run-time t_1 of A_1 plus the time $q_d \cdot O(t_F + q_k)$ to search KRA queries and run F for each designation query of A_1 . All other queries of A_1 are simply forwarded by \widehat{A}_1 to its oracles. This completes the proof. \square

A.2 Proof of Theorem 3

Proof of (1). We show that the UDVS scheme DVS constructed from the given separable EK ID-KEM scheme KEM as in Section 5 has all the claimed properties.

Consistency: PV Verifiability. For any $(sk_1, pk_1) = \text{GKS}(cp)$, we have that $sk_1 = (cp_1, cp_2, mk)$. So $\sigma_{pv} = S(sk_1, m) = \text{Extract}((cp_1, cp_2), mk, m)$ is the user secret-key corresponding to user identity m and hence $\widehat{K}' = \text{Decrypt}((cp_1, cp_2), \sigma_{pv}, \text{Enc}_c((cp_1, cp_2), m; \widehat{r}))$ in V is equal to $\widehat{K} = \text{Enc}_K((cp_1, cp_2), m; \widehat{r})$ by consistency of KEM, so V returns *Acc*.

Consistency: DV Verifiability. From the definition of GKV we have that $pk_3 = \text{Enc}_c((cp_1, cp_2), ID_0; sk_3)$, and using the Separable and EK properties of KEM, we also have $pk_3 = \text{Enc}_c((cp_1, cp_2), m; sk_3)$ for any m . So since $\sigma_{pv} = S(sk_1, m) = \text{Extract}((cp_1, cp_2), mk, m)$ is the user secret-key corresponding to identity m , it follows from the consistency of KEM that $\widehat{\sigma}_{dv} = K_{c,m} = \text{Decrypt}((cp_1, cp_2), \sigma_{pv}, \text{Enc}_c((cp_1, cp_2), m; sk_3))$ is equal to $\widehat{K}_{c,m} = \text{Enc}_K((cp_1, cp_2), m; sk_3)$ so VDV returns *Acc*.

DVSig-Uniqueness. Given (cp_1, pk_1, pk_3, m) , the DV signature $\sigma_{dv} = \text{Decrypt}_{pk_1, \sigma_{pv}, pk_3}$ is uniquely determined by (cp_1, pk_1, pk_3, m) since σ_{pv} is the secret-key corresponding to identity m and all secret-keys corresponding to a given identity must give identical decryptions of any given ciphertext, to satisfy the consistency of KEM.

DV-Unforgeability. Given any efficient DV-UF attacker A against DVS with resources (t, q_s) and non-negligible success probability $\text{Succ}_{A, \text{DVS}}^{\text{UF-DV}}(k)$, we construct an efficient ST-ID-OW attacker \widehat{A} against KEM, which works as follows on input (cp_1, cp_2) . Let $ID_0 \in S_{ID}$ be any identity string. In Stage 1, \widehat{A} just outputs ID_0 as the challenge identity. In Stage 2, \widehat{A} is given the challenge ciphertext $c^* = \text{Enc}_c((cp_1, cp_2), ID_0; r^*)$ for uniformly random $r^* \in S_R$. Then \widehat{A} sets $pk_3 = c^*$, $pk_1 = (cp_1, cp_2)$ and runs A on input (cp_1, pk_1, pk_3) . When A queries a message m_i to its S oracle, \widehat{A} forwards it to its Extract oracle and returns the answer to A . Eventually, A outputs a forgery (m^*, σ_{dv}^*) , for a new message m^* never queried by A to be signed and hence never queried by \widehat{A} to Extract . Then \widehat{A} outputs (m^*, σ_{dv}^*) as its ID/decrypted-key solution pair. Since \widehat{A} simulated the view of A exactly as in a UF-DV attack, we know that, with probability at least $\text{Succ}_{A, \text{DVS}}^{\text{UF-DV}}(k)$, we have $\sigma_{dv}^* = \text{Enc}_K(cp_1, cp_2, m^*; r^*)$, which by consistency of KEM is equal to $\text{Decrypt}(cp_1, cp_2, sk_{m^*}, \text{Enc}_c(cp_1, cp_2, m^*; r^*))$, which in turn is equal to $\text{Decrypt}(cp_1, cp_2, sk_{m^*}, c^*)$ by the EK property of KEM, which is the desired output for \widehat{A} (here sk_{m^*} is the secret key corresponding to identity ID).

So \hat{A} breaks ST-ID-OW with non-negligible probability $\text{Succ}_{A,DVS}^{\text{UF-DV}}(k)$, with efficient running time t , and q_s extraction queries, contradicting the assumed ST-ID-OW security of KEM.

Complete Unconditional Privacy. We show the existence of an efficient universal forgery algorithm F , which on input $(cp_1, pk_1, (sk_3, pk_3), m^*)$, computes the unique DV signature $\sigma_{dv} = \text{CDV}(pk_1, pk_3, m^*, S(sk_1, m^*))$ with probability 1. The claimed complete and unconditional privacy then follows by applying Lemma 1. The forger F computes the forgery as in the DV verification algorithm, i.e. $\hat{\sigma}_{dv} = \text{Enc}_K(pk_1, m^*; sk_3)$. The algorithm is efficient and is correct with probability 1 due to perfect consistency of KEM, as shown in the proof of the DV-Consistency property.

This completes the proof of part (1).

Proof of (2). We show that the UDVS scheme DVS constructed from the given separable EK ID-KEM scheme KEM as in Section 5 has all the claimed properties.

Consistency. By the assumed privacy of DVS we have from Lemma 1 that the encrypted key $K = F(cp_1, pk_1, sk_3, pk_3, ID)$ is equal to the decrypted key $K' = \text{CDV}(pk_1, pk_3, ID, S(sk_1, ID))$ with probability 1, so KEM is consistent.

ST-ID-OW Security. Given any efficient ST-ID-OW attacker A against KEM with resources (t, q_E) and non-negligible success probability $\text{Succ}_{A,KEM}^{\text{ST-ID-OW}}(k)$, we construct an efficient UF-DV attacker \hat{A} against DVS, which works as follows on input (cp, pk_1, pk_3) . First, \hat{A} runs A on input $cp = (cp, pk_1)$. When A queries an identity ID_i to its Extract oracle, \hat{A} forwards it to its S oracle and returns the answer to A . At the end of its Stage 1, A outputs a challenge identity ID , and \hat{A} returns the ciphertext pk_3 to A . At the end of Stage 2, A outputs a solution (\hat{ID}, K') , and \hat{A} outputs (\hat{ID}, K') as its message/DV sig. forgery pair. Since \hat{A} simulated the view of A exactly as in a ST-ID-OW attack, we know that, with probability at least $\text{Succ}_{A,KEM}^{\text{ST-ID-OW}}(k)$, \hat{A} 's output is equal to decrypted key $\text{Decrypt}(cp, sk_{\hat{ID}}, pk_3)$ for ciphertext pk_3 with respect to identity \hat{ID} , namely the unique DV Sig. $\sigma_{dv}^* = \text{CDV}(pk_1, pk_3, \hat{ID}, S(cp, sk_1, \hat{ID}))$ on message \hat{ID} , which was not queried by A to Extract, and thus not queried by \hat{A} to S . So \hat{A} breaks UF-DV of DVS with probability $\text{Succ}_{A,KEM}^{\text{ST-ID-OW}}(k)$, running time t , and q_E signature queries. This completes the proof of part (2). \square

Author Index

Al-Riyami, Sattam S. 452
Ateniese, Giuseppe 246
Attrapadung, Nuttapong 374

Bauer, Mark L. 311
Billet, Olivier 331
Biryukov, Alex 228
Bresson, Emmanuel 37
Bull, Laurence 523

Catalano, Dario 37
Chang, Donghoon 208
Clarke, Dwaine 188
Coron, Jean-Sebastien 392

Devadas, Srinivas 188
Dijk, Marten van 188
Dodson, Bruce 55
Duursma, Iwan 111

Fouque, Pierre-Alain 492

Gassend, Blaise 188
Gilbert, Henri 331
Gower, Jason E. 302

Hamdy, Safuat 311
Howgrave-Graham, Nick 492
Huang, Hsing-Hui 326
Hughes, James 55
Hwang, Yong Ho 359

Imai, Hideki 155, 374

Kim, Chong Hee 359
Kobara, Kazukuni 155, 374
Kohel, David R. 124
Kortsmit, Wil 55
Kurosawa, Kaoru 19, 474

Lee, Hyang-Sook 111
Lee, Pil Joong 359
Lee, Sangjin 208
Lee, Wonil 208
Lenstra, Arjen 55
Leyland, Paul 55
Lipmaa, Helger 398, 416

Martinet, Gwenaëlle 492
Medeiros, Breno de 246
Mihaljević, Miodrag J. 137
Mishra, Pradeep Kumar 93
Muller, Frédéric 347

Naccache, David 392
Nandi, Mridul 208

Okamoto, Tatsuki 287

Paterson, Kenneth G. 452
Phan, Duong Hieu 1
Pieprzyk, Josef 507, 523
Pointcheval, David 1, 37
Poupard, Guillaume 492
Preneel, Bart 228

Rompay, Bart Van 228

Sarkar, Palash 93
Sato, Hisayoshi 434
Schmidt-Samoa, Katja 474
Shamir, Adi 55
Shin, SeongHan 155
Steinfeld, Ron 523
Stern, Jacques 287
Suh, G. Edward 188
Sung, Soohak 208

Takagi, Tsuyoshi 19, 434, 474
Takaragi, Kazuo 434
Tezuka, Satoru 434
Thériault, Nicolas 75
Tromer, Eran 55
Tsudik, Gene 269

Vandewalle, Joos 228

Wang, Chih-Hung 173
Wang, Huaxiong 507, 523

Xu, Shouhuai 269

Yan, Hong-Sen 326